HERRAMIENTA PARA EL ENTRENAMIENTO EN LA CONSTRUCCIÓN Y EVALUACIÓN DE EXPRESIONES ARITMÉTICAS LÓGICAS Y CONDICIONALES

FRANKLIN ACOSTA MATEUS

CAMILO ANDRES RODRÍGUEZ MARTINEZ

NELSON GABRIEL CÁRDENAS URUEÑA

UNIVERSIDAD PILOTO DE COLOMBIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ D.C
2017

HERRAMIENTA PARA EL ENTRENAMIENTO EN LA CONSTRUCCIÓN Y EVALUACIÓN DE EXPRESIONES ARITMÉTICAS LÓGICAS Y CONDICIONALES

FRANKLIN ACOSTA MATEUS

CAMILO ANDRES RODRÍGUEZ MARTINEZ

NELSON GABRIEL CÁRDENAS URUEÑA

Proyecto de grado para optar por el título de Ingeniero de Sistemas

Asesores

Ignacio Hernández Molina
Sociólogo

Giovanni Fajardo Utria
Ingeniero de Sistemas

UNIVERSIDAD PILOTO DE COLOMBIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ D.C
2017

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del director del proyecto

_____
Firma del jurado

_____
Firma del jurado

_____
Firma del jurado

Bogotá D.C, 15 de noviembre de 2017

DEDICATORIA

Queremos dedicar este trabajo a nuestros padres por la paciencia y esfuerzo otorgado durante tantos años de estudio, a las personas que estuvieron de la mano en nuestros problemas e incertidumbres, a nuestros amigos y compañeros que estuvieron con nosotros apoyándonos, aconsejándonos y alentándonos cada día para seguir adelante.

AGRADECIMIENTOS

Queremos agradecer especialmente a nuestro director Giovanni Fajardo Utria que, con su paciencia, dedicación y apoyo, permitió hacer de este proyecto un gran aporte no solo a nuestra formación profesional, si no a la comunidad de estudiantes del área de ingeniería de sistemas. A los docentes Ignacio Hernández Molina y demás involucrados en la carrera de la facultad de sistemas quienes nos apoyaron e hicieron parte de la elaboración de esta investigación. A nuestros padres, por su apoyo incondicional y por confiar en nosotros como futuros profesionales en Ingeniería de sistemas, sin ellos este sueño no sería una realidad. A la Universidad Piloto de Colombia por el espacio de cada una de sus instalaciones, para cumplir una meta más en nuestra vida profesional.

CONTENIDO

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE GRAFICAS

pág.

GLOSARIO

ABSTRACCIÓN: es el énfasis en el "¿Qué hace?" más que en el "¿Cómo lo hace?".

ACTOR: es una entidad externa al sistema que se modela y que puede interactuar con él.

AJAX: es una técnica de desarrollo web para crear aplicaciones interactivas que se ejecutan en el cliente o navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano, por lo que se pueden realizar cambios sobre las paginas sin necesidad de recargarlas, mejorando la interactividad, velocidad, y usabilidad en la aplicación.

ALGORITMO: es un conjunto finito de instrucciones bien definidas en su lógica de control que permiten la solución de un problema en una cantidad finita de tiempo.

ANGULARJS: es un framework estructural de código abierto y gratuito desarrollador por Google, está basado en el lenguaje JavaScript y tiene como objetivo principal crear aplicaciones web dinámicas y eficientes, enfocándose únicamente en administrar la parte lógica de la aplicación.

ASP.NET Web API: es un marco que facilita la creación de servicios HTTP disponibles para una amplia variedad de clientes, entre los que se incluyen exploradores y dispositivos móviles, Además es una plataforma perfecta para crear aplicaciones RESTful .NET Framework.

CASOS DE USO: es un conjunto descriptivo de situaciones que ocurren entre el usuario (Actor) y la simulación de la aplicación en sí.

COMPILADOR: es un programa que lee un programa escrito en un lenguaje, el lenguaje fuente, y lo traduce a un programa equivalente en otro lenguaje, el lenguaje objeto. Como parte importante de este proceso de traducción, el compilador informa a su usuario de la presencia de errores gramáticos, sintáctico y semánticos en el programa fuente.

CCS3: es lenguaje en su versión número 3 que sirve para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML.

C#: es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.

DIAGRAMA DE ACTIVIDAD: es un diagrama de flujo que modela las acciones que el objeto va a realizar, y en qué orden.

DIAGRAMA DE CASOS DE USO: es un modelamiento de casos de uso que muestra los tipos de interacciones que tienen los usuarios con un sistema.

DOCUMENTO DE CASOS DE USO: documento que contiene la especificación de casos de uso definidos para un sistema. Así mismo, puede contener un diagrama de casos de uso.

DIAGRAMA DE FLUJO: es la representación gráfica mediante símbolos especiales, de los pasos o procedimientos de manera secuencial y lógica que se deben realizar para solucionar un problema dado.

FRAMEWORK: es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Además, pueden incluir soporte de programas, bibliotecas, entre otros.

HTML5: es un lenguaje de etiquetas en su versión número 5 que permite construir documentos webs de forma que los navegadores puedan entender el contenido y mostrárselo al usuario.

HTTP: HyperText Transfer Protocol (Protocolo de transferencia de hipertexto) es el método más común de intercambio de información en la world wide web.

IDENTIFICADORES: son los nombres que se les asignan a los objetos, los cuales se pueden considerar como variables o constantes, éstos intervienen en los

procesos que se realizan para la solución de un problema, por consiguiente, es necesario establecer qué características tienen.

JAVASCRIPT: es un lenguaje interpretado, basado en prototipos que no requiere de compilación ya que funciona del lado del cliente, por lo tanto, los navegadores web son quienes interpretan este código.

JQUERY: es un framework de JavaScript permite el acceso a los elementos del DOM, los efectos, interactuar con los documentos HTML, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

JSON: Acrónimo de JavaScript Object Natation, es un formato de texto ligero para el intercambio de datos, y es considerado un formato de lenguaje independiente debido a su amplia adopción como alternativa a XML.

LENGUAJE DE PROGRAMACIÓN: conjunto de reglas y estándares que es utilizado para escribir programas de computador (software), que puedan ser entendidos por él.

LÓGICA: es una ciencia formal que se basa en leyes, modalidades y formas del conocimiento científico, es decir, que propone estudiar los métodos y principios adecuados para identificar el razonamiento correcto frente al que no lo es.

PROGRAMACIÓN: es el proceso de diseñar, codificar, depurar y mantener el código fuente de programas computacionales.

PROGRAMA: es la representación de algún software en un lenguaje de programación específico.

PSEUDOCÓDIGO: es una de las herramientas más conocidas para el diseño de solución de problemas por computadora. Permite pasar casi de manera directa la solución del problema a un lenguaje de programación específico.

WEB RESPONSIVE DESIGN: es una técnica de diseño y desarrollo web que mediante el uso de estructuras, imágenes fluidas y media-queries en la hoja de estilo CSS, consigue adaptar el sitio web al entorno del usuario.

RESTFUL: son web services que siguen los principios principales de rest entre los cuales se encuentra; saber que todo lo que se mueve a través de las comunicaciones web es un recurso que debe tener un identificador único y múltiples representaciones, Además este protocolo de transmisión de datos debe utilizar los verbos estándares de HTTP.

SOFTWARE: conjunto de instrucciones que incluyen datos, procedimientos y pautas que permiten realizar diferentes tareas en un sistema computacional donde le dicen al hardware que hacer.

SQLSERVER: es un sistema de bases de datos profesional de Microsoft, que contiene una variedad de características y herramientas que se pueden utilizar para desarrollar y administrar bases de datos. Además de almacenar, procesar y proteger los datos.

RESUMEN

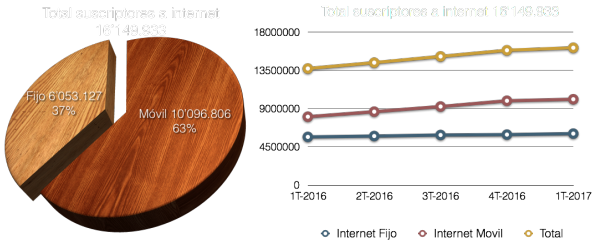
Debido a la carencia de instrumentos para el entrenamiento en la construcción e interpretación de expresiones aritméticas lógicas y condicionales en la Universidad Piloto De Colombia, se desarrolló una herramienta para que el estudiante pueda entrenarse en los conceptos básicos de programación desde cualquier lugar. Para ello, se tuvo en cuenta los diferentes dispositivos móviles a los que se pretendía llegar, además de la disposición junto con el mantenimiento que se le puede brindar a la herramienta para que cada estudiante o usuario de la misma tenga las ultimas y más recientes características en su dispositivo. Donde finalmente, se evaluó e implementó las tecnologías y estructuras más adecuadas para cumplir con los objetivos propuestos y funcionamiento crítico de la aplicación.

INTRODUCCIÓN

El nuevo sistema universitario requiere modificar la metodología y la relación docente-alumno, en cuanto al rol que ambos desempeñan, el docente debe ser más un guía que facilite el aprendizaje, apoyándose en diferentes recursos, mientras que el estudiante debe asumir un papel más activo y responsable en su proceso formativo. Hoy en día podemos encontrar diversas herramientas que son moderadamente utilizadas por el alumno en el ámbito universitario, teniendo en cuenta que las mismas pueden llevar a mejorar significativamente el proceso de enseñanza y aprendizaje[1].

Las herramientas denominadas como "herramientas web" suponen un nuevo paradigma sobre el diseño y uso de internet, al permitir crear redes de interacción y comunicación en línea, haciendo que el internet, sea un lugar tanto para leer información como para escribir sobre él. Este cambio tecnológico lleva consigo un cambio de actitud en los estudiantes, ya que fomentan su participación en la creación de nuevo contenido y se permite interactuar y aprender junto con él, propiciando con ello a desarrollar nuevas capacidades y competencias[2], donde según las siguientes estadísticas al cierre del primer trimestre del 2017, se reporta que el número total de suscriptores a Internet estaba compuesto principalmente por accesos móviles, con 10.096.806 suscripciones y una participación del 63%; mientras que los suscritores fijos a Internet alcanzaron los 6.053.127 y una participación del 37%[3].

Grafica 1. Conexiones de internet Grafica 2. Crecimiento conexiones de internet



[1] GARCÍA, Jose Luis y GARCÍA, Rosa. Aprendizaje entre Iguales con Herramientas Web 2.0 y Twitter en la Universidad. Análisis de un Caso. En: Revista Electrónica de Tecnología Educativa. Junio, 2012, Vol. 40, p. 1-14

[2] VAQUERIZO, Belén, RENEDO, Eduardo y VALERO, Miguel. Aprendizaje colaborativo en grupo: Herramientas Web 2.0. En: XV JENU. Julio, 2009, p. 447-450.

[3] MINTIC, (Suscriptores a internet fijo y móvil en: Boletín trimestral de las TIC) [en línea]. (Colombia): Ministerio de tecnologías de la información y las comunicaciones Julio, 2017- [citado 1 de septiembre], disponible desde: http://colombiatic.mintic.gov.co/602/w3-article-55212.html.

Así bien, con el surgimiento de la World Wide Web que ha provocado el crecimiento exponencial de la información, la comunidad científica se ha preocupado por la proliferación de aplicaciones y programas de software para el aprendizaje en el internet, su relación, su impacto y sus efectos en el proceso enseñanza/aprendizaje; Encontrando que diversos aplicativos pueden propiciar en el estudiante su aprendizaje de acuerdo con sus intereses y sus destrezas cognitivas[4].

Ante este panorama y dada la problemática, esta investigación se centra en el desarrollo de una herramienta Web aplicada a la educación, ya que la UPC carece de instrumentos que faciliten el entrenamiento, desarrollo y codificación de código a través de un dispositivo que esté al alcance del estudiante.

[4] CELA, Karina. et al. Evaluación de herramientas web 2.0, estilos de aprendizaje y su aplicación en el ámbito educativo. En: Revista Estilos de Aprendizaje. Abril, 2010, Vol. 3 N.5, p. 117-134

1. JUSTIFICACIÓN

En la UPC se evidencia la falta de aplicaciones, proyectos o soluciones que faciliten programar en cualquier momento y lugar, generando poca motivación e interés en esta rama, debido a esto, se pretende desarrollar una herramienta web la cual permitiría a usuarios, sean estos estudiantes y/o personas externas, entrenar su construcción y evaluación de expresiones lógicas, aritméticas, condicionales y ciclos.

2. PROBLEMA

2.1 DESCRIPCIÓN DEL PROBLEMA

En la Universidad Piloto de Colombia (UPC) se evidencia que los estudiantes del programa de Ingeniería de Sistemas carecen de herramientas para el entrenamiento en la construcción de programas básicos utilizando las primitivas básicas de programación, tales como: escribir, leer, asignación, expresiones lógicas y aritméticas, condicionales y ciclos; que son primordiales para programar, siendo este uno de los ejes fundamentales de la carrera.

Esta carencia conlleva a una inclinación apática a la programación por parte de los estudiantes ya que no es fácil conseguir herramientas que sean accesibles desde su usabilidad y portabilidad, provocando que su interés se vea disminuido. Por lo tanto, las metodologías, herramientas o propuestas didácticas que se ofrecen en los cursos de programación presentan dificultades, las cuales hacen que los estudiantes desde el inicio de la carrera hasta su culminación se vean afectados.

2.2 FORMULACIÓN DEL PROBLEMA

¿Cómo desarrollar una herramienta que ayude al entrenamiento de las primitivas básicas de programación?

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Desarrollar una herramienta que ayude en la construcción e interpretación de las primitivas básicas de programación, desde cualquier dispositivo.

3.2 OBJETIVOS ESPECÍFICOS

- Identificar cual es la estructura sintáctica y gramatical de las primitivas básicas de programación.
- Realizar un proyecto web responsive design para acceder a un editor de código en bloque desde cualquier dispositivo.
- Desarrollar un teclado que proporcione las estructuras sintácticas de las primitivas básicas de programación.

4. LIMITES

La presente “Herramienta para el entrenamiento en la construcción y evaluación de primitivas básicas de programación” hace parte de un proyecto macro que está orientado desde la perspectiva del proyecto Ariadna para el programa de Ingeniería de Sistemas de la UPC, por consiguiente, esta propuesta solo se va a enfocar en el entrenamiento para los estudiantes de primeros semestres (1 y 2) que comienzan con la programación en la carrera de ingeniería de Sistemas y Telecomunicaciones.

Además, esta herramienta no cuenta con programación orientada a objetos ni interfaz gráfica, solo imprimiendo resultados por consola.

5. ALCANCE

Se espera que el estudiante que se inicia en la lógica de la programación cuente con la herramienta que se propone en el presente proyecto de grado, para poder entrenar, fortalecer y mejorar su desempeño en cuanto a la programación básica ayudando al estudiante a evitar errores sintácticos y gramaticales.

6. SISTEMA DE HIPÓTESIS

6.1 HIPÓTESIS DE TRABAJO

La implementación de una herramienta web permitirá la construcción e interpretación de primitivas básicas de programación.

6.2 HIPÓTESIS NULA

La implementación de una herramienta web no permitirá la construcción e interpretación de primitivas básicas de programación.

7. SISTEMA DE VARIABLES

7.1 VARIABLE INDEPENDIENTE

- Herramienta web.
- Lenguaje de programación.
- Metodologías complejas.

7.2 VARIABLE DEPENDIENTE

- Entrenamiento en construcción de primitivas básicas de programación.
- Programación.
- Problemas para la construcción de primitivas básicas de programación por parte de los estudiantes.

7.3 VARIABLE INTERVINIENTE O EXTERNA

- Competencias en programación.
- Estudiantes.
- Escalabilidad, reutilización.
- Conocimientos previos

8. MARCO TEÓRICO

Para entender un poco más la problemática del presente estudio se requiere el conocimiento de diversas definiciones frecuentemente utilizadas en ingeniería, las cuales son presentadas a continuación:

8.1 LÓGICA

Viene de la palabra "*logos* (razón o palabra), donde es en realidad el estudio de la razón en vistas de su dirección adecuada"[5], además se entiende como la disciplina que trata de la formulación de los métodos de investigación científica y le corresponde "analizar los procesos del pensamiento para descubrir las formas que adoptan los elementos del pensamiento, las funciones que los enlazan, los métodos empleados en la investigación y las leyes del conocimiento teórico y experimental". De esta manera la lógica está en constante búsqueda de la demostración e inferencia valida, pues tiene su fundamento en el conocimiento científico[6].

8.1.1 Clasificación de la lógica. Dos tipos de lógica han surgido de las dos fases principales en el desarrollo de la ciencia lógica, que son:

8.1.1.1 Lógica Dialéctica. La lógica dialéctica no constituye un cuerpo teórico organizado y estructurado, como en la lógica matemática, ni tampoco una serie de estrategias discursivas para el desarrollo de argumentaciones, como lo es la lógica informal. Se trata realmente de una serie de formulaciones filosóficas heterogéneas cuyos temas centrales como el tiempo y el cambio de los procesos[7].

[5] BEUCHOT, Mauricio. Introducción a la lógica. México D.F: Universidad Autónoma de México. 1era edición, 2004. 175p.

[6] LEFEBVRE, Henri. Lógica formal, Lógica dialéctica. España: Siglo XXI Editores. 1era Edición, 1993. 346p.

[7] BELLER, Walter. Érase un tópico en filosofía: la lógica dialéctica. En: Revista Casa del tiempo. Julio-Agosto, 2015, Vol. 2, N 18-19, p. 32-34.

8.1.1.2 Lógica Formal. La lógica formal fue el primer gran sistema del conocimiento científico de los procesos del pensamiento[8]. Es considerada como la disciplina que hace abstracción del desarrollo y las transformaciones de los procesos de la realidad[6]. El principal objetivo de la lógica formal es proporcionar un punto de referencia para distinguir argumentos válidos de los que no son válidos. Además, el método más común de ésta es la deducción, que intenta establecer la verdad de sus conclusiones a través de dos condiciones: la conclusión tiene que emanar de las premisas y las premisas tienen que ser ciertas. Si se cumplen las dos, se dice que el argumento es válido[9].

8.1.1.3 Cálculo formal. Es un sistema lingüístico que consta de un conjunto de símbolos elementales, una serie de reglas de formación para la combinación de dichos símbolos elementales y un conjunto de reglas de transformación para pasar de una combinación de símbolos a otra. Además, su característica radica en la desconexión semántica de los símbolos, lo cual indica que no tiene en cuenta las significaciones o aplicaciones que de él se puedan extraer[10].

8.1.1.4 Importancia de la lógica Dialéctica y Formal. Es necesario mencionar la lógica dialéctica y la lógica formal, y a la misma vez hacer una disimilitud entre ambas. La lógica dialéctica " afina y aumenta nuestra capacidad de lograr una comprensión más profunda y clara de la realidad" la cual es la que acompaña al entendimiento en todo su proceso de pensamiento, ayudándolo a crear imágenes mentales de las realidades dadas, creando inferencias, formulando juicios, y es la mediadora de la lógica formal la cual es entendida como "la ciencia que estudia las modalidades del pensamiento correcto, en las cuales se reflejan las relaciones más simples que existen en los procesos" la lógica formal es el dominio de las operaciones formales y del pensamiento correcto, representa los procesos como objetos y "nos enseña cómo se utilizan los conceptos, los juicios y las inferencias para pensar de un modo ordenado, preciso, coherente, consecuente y riguroso"; estas características, tienen estrecha similitud con las de los algoritmos, y es por ello que es indispensable hablar de ellos[5].

---

[8] NOVACK, George. Introducción a la lógica, lógica formal y lógica dialéctica. España: Editorial Fontamara, S.A. 1era Edición, 1979. 74p.

[9] WOODS, Alan y GRANT, Ted. La lógica formal y la dialéctica. En: Razón y Revolución. Reedición Electrónica, 2002, No 10, p. 1-27.

[10] VELARDE, Julián. Lógica y Dialéctica 1. En: Teorema: International Journal of Phylosophy, 1977, Vol 7, No 2, p. 129-140.

8.2 HERRAMIENTAS

Para implementar la solución de un problema mediante el uso de una computadora es necesario establecer una serie de pasos que permitan resolver el problema, a este conjunto de pasos se le denomina algoritmo[11].

8.2.1 Algoritmo. Es un conjunto finito de instrucciones bien definidas en su lógica de control que permiten la solución de un problema en una cantidad finita de tiempo. El algoritmo realiza un conjunto de pasos cuya ejecución para dar la solución del problema puede ser ejecutada manualmente, mecánicamente o utilizando una máquina de procesamiento electrónico de datos[12].

Teniendo como fundamentos que la lógica es entendida como un conjunto de reglas que conforman un sistema de razonamiento y, a su vez, siendo el algoritmo un conjunto finito de reglas bien conformadas en su lógica de control y factibles de ser automatizadas mediante un lenguaje de programación, el análisis de la relación entre algoritmo y lógica se basará en la Teoría de la Información; justificado por el hecho de que es precisamente la operación de la lógica de control del algoritmo la que permite el procesamiento de los datos para generar información y, en ultimas, conocimiento útil en el contexto de producción de un sistema informático[6].

8.3 PENSAMIENTO ALGORÍTMICO

Cuando se habla de algoritmos, con frecuencia aparecen tres tipos de pensamiento que generalmente se relacionan con ellos y que se utilizan indiscriminadamente como sinónimos: Pensamiento Computacional, Pensamiento Algorítmico y Pensamiento Procedimental, lo cual hace importante puntualizar a qué se refiere cada uno de estos pensamientos[12].

8.3.1 Pensamiento Computacional. Hace referencia a la representación y solución de problemas utilizando inteligencia humana, de máquinas o de otras formas que ayuden a resolver el problema.

---

[11] DELGADO, Francisco y AMADOR, César. Algoritmos resueltos con diagramas de flujo y pseudocódigo. México: Universidad Autónoma de Aguascalientes. 1era Edición, 2014. 170p.

[12] MANCILLA, Alfonso, EBRATT, Jesús y CAPACHO, José. Diseño y construcción de algoritmos. Colombia: Editorial Universidad del Norte. 2014. 414p.

8.3.2 Pensamiento Algorítmico. Se refiere al desarrollo y uso de algoritmos que puedan ayudar a resolver un tipo específico de problema o a realizar un tipo específico de tarea.

8.3.3 Pensamiento Procedimental. Se ocupa del desarrollo y utilización de procedimientos diseñados para resolver un problema específico o para realizar un tipo específico de tarea, pero que no siempre resulta exitoso.

El pensamiento Algorítmico está fuertemente ligado al pensamiento procedimental requerido en la programación de computadores. Sin embargo, su desarrollo puede conducir a los estudiantes a aproximarse guiada y disciplinadamente a los problemas de forma que este pueda transferirse a otros ambientes diferentes a los de la programación[13].

Tabla 1. Características de un Algoritmo

PRECISIÓN	DETERMINISMO	FINITUD
Orden preciso en el cual deben ejecutarse las tareas que conforman el algoritmo[14].	Todas las veces que se realicen las tareas o pasos de un algoritmo, con las mismas condiciones iniciales, se deben obtener resultados idénticos[13].	El algoritmo debe terminar en algún momento y debe usar una cantidad finita de recursos[13].

Programación de computadores

8.4 PROGRAMACIÓN

La primera dificultad del aprendizaje de la programación radica en la necesidad de aprender dos cosas bastante diferentes de manera simultánea: La primera, es un lenguaje para transmitir a la máquina las órdenes que se le quiere dar, es decir, el lenguaje de programación y una manera de pensar y concebir órdenes a la computadora; y segundo, el algoritmo traducido en programa[14].

[13] LÓPEZ, Juan Carlos. Algoritmos y Programación. En: Fundación Gabriel Piedrahita Uribe. 2da Edición, 2009. 96p.

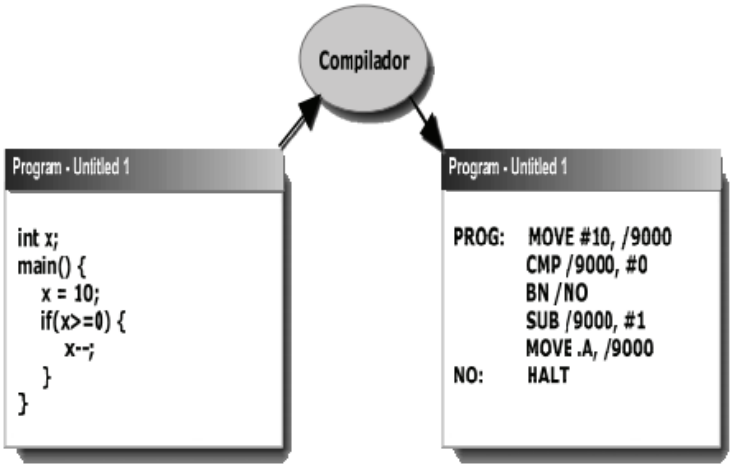
[14] PROGRAMACIÓN DE COMPUTADORES. Publicación Facultad de Ingeniería, Universidad Nacional de Colombia, 2004.

La Programación no es solo el trabajo de escritura del código, sino todo un conjunto de tareas que se deben cumplir, a fin de que el código que se escribió resulte correcto y robusto, y cumpla con el objetivo o los objetivos para los que fue creado[15].

Si un programa está escrito en un lenguaje de programación comprensible para el ser humano, se le llama código fuente. A su vez, este código se puede convertir en un archivo ejecutable con la ayuda de un compilador, aunque también puede ser ejecutado a través de un intérprete[14].

8.4.1 Compilador. Es un programa que lee un programa escrito en un lenguaje, el lenguaje fuente, y lo traduce a un programa equivalente en otro lenguaje, el lenguaje objeto. Como parte importante de este proceso de traducción, el compilador informa a su usuario de la presencia de errores en el programa fuente[16].

Figura 1. Esquema de un Compilador[17].



Teoría e Implementación

[15] JUGANARU MATHIEU, Mihaela. Introducción a la programación. México D.F: Grupo editorial patria, S.A de C.V. 1era Edición, 2014. 320p.

[16] AHO, Alfred. et al. Compiladores: principios, técnicas y herramientas. 2da Edición. México: Pearson Educación, 1990. 820p.

[17] RUIZ CATALÁN, Jacinto. COMPILADORES: Teoría e Implementación. España: RC Libros, 2010. 16p.

8.5 ESTRUCTURAS DE CONTROL

Normalmente un programa cuenta con instrucciones que se ejecutan una tras de otra, es decir en el orden que están escritas, a este proceso se le denomina ejecución secuencial, y las instrucciones que el programador especifique ejecutarse así no sea la que sigue en la secuencia se le denomina transferencia de control[18].

La programación estructurada fue tomada enserio por parte de los desarrolladores aproximadamente en el año de 1970 ya que los resultados obtenidos eran positivos, puesto que los programas eran más claros, mas fáciles de depurar y modificar, además que desde un comienzo estos estaban libres de errores, de esta manera se reportaba reducción en tiempos de desarrollo, mayor incidencia de entregas de sistemas y más proyectos de software finalizados sin salirse del presupuesto[17].

Todos los programas podían escribirse en términos de tres estructuras de control solamente: la estructura de secuencia, la estructura de selección y la estructura de repetición, siendo esto demostrado por Bohm y Jacopi, eliminando la instrucción goto que en 1960 genero dificultades en el desarrollo de software puesto que permitía al programador especificar la transferencia de control a uno de los muchos posibles destinos dentro de un programa[17].

8.5.1 Estructura de secuencia en Java. La estructura de secuencia está integrada en Java, a menos que se le indique lo contrario la computadora ejecuta las instrucciones en Java una después de la otra, en el orden que estén escritas[17].

8.5.2 Instrucciones de selección en Java. if es una instrucción de selección simple ya que selecciona o ignora una sola acción. if else es una instrucción de selección doble, ya que selecciona entre dos acciones distintas y switch es una estructura de selección múltiple, ya que selecciona entre diversas acciones[17].

8.5.3 Instrucciones de repetición en Java. Las instrucciones while y for realizan la acción en sus cuerpos, cero o más veces; si en un principio la condición de continuación del ciclo es falsa, no se ejecutará la acción. Mientras que la instrucción do while realiza la acción en su cuerpo, una o más veces[17].

[18] DEITEL, Harvey y DEITEL, Paul. Como programar en java. México: Pearson Educación. 9ª Edición, 2012, 616p.

9. METODOLOGÍA

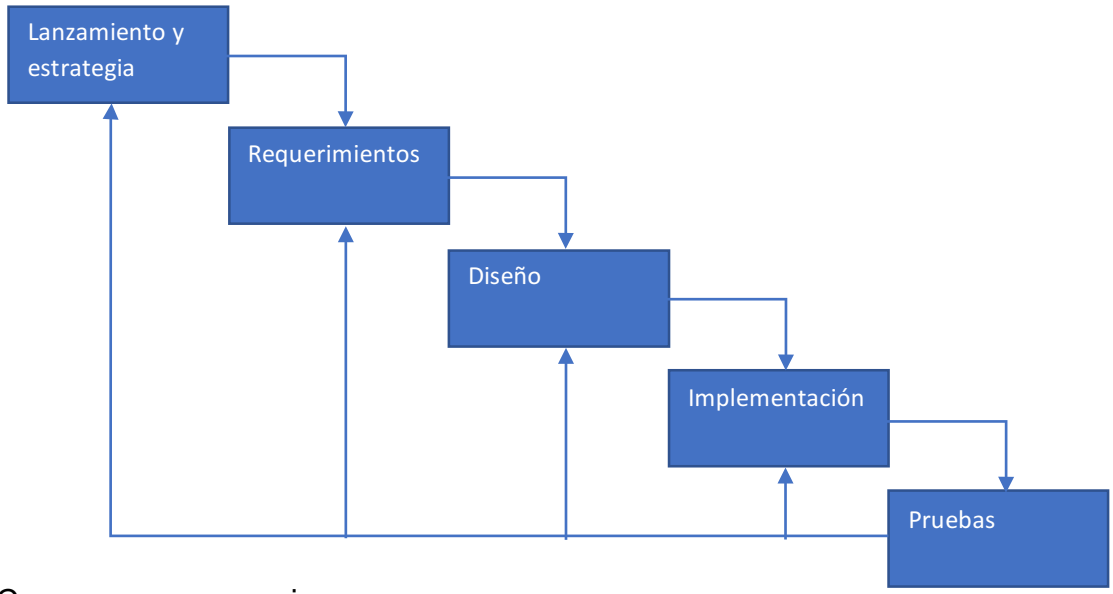
9.1 DISEÑO Y DESARROLLO METODOLÓGICO

La elaboración de la aplicación web será desarrollada con un modelo en cascada[19], debido a que permite tener un control satisfactorio en cuanto a organización y estandarización en cada una de sus fases de creación de la herramienta.

Se realizará una evaluación del entorno móvil en el cuál se ejecutará la herramienta web, ya que de esta manera se logrará tener una idea global para elegir el lenguaje más adecuado para el desarrollo de la aplicación.

La manera en que se desarrollara la aplicación se compone de diferentes fases:

Figura 2. Modelo en cascada[17].



Como programar en java

9.1.1 Fase de lanzamiento y estrategia. En esta fase de lanzamiento y estrategia establecemos unos objetivos, estrategias y demás conceptos y así hacer una planificación más detallada que mencionaremos a continuación para la realización del producto(Teseo).

[19] SOMMERVILLE, Ian. Ingeniería del software. España: Pearson Educación, S.A. 7ª Edición, 2005. 712p.

9.1.1.1 Objetivo

- Obtener una estrategia eficaz que permita desarrollar un software de alta calidad y de gran utilidad.
- Aumentar la planificación y gestión del proyecto para mitigar errores del software al concluir con el alcance del proyecto.
- Especificar el orden en el cual las funciones del producto son definidas, diseñadas, implementadas y probadas.

9.1.1.2 Estrategia

- Comprensión general del equipo acerca del trabajo a desarrollar.
- Estimar de manera preliminar el tamaño y tiempo de desarrollo del proyecto en desarrollarse.
- Identificar principales funcionalidades del software que se van a implementar.

9.1.1.3 Criterios de la estrategia

- Establecer una base para hacer la planeación detallada.
- Documento dirigido a los que estén directamente involucrados indicando de manera general las características del producto a desarrollar.
- Orden en que se definen, diseñan e implementan las funciones o módulos del producto.

9.1.1.4 Estrategia general. Desarrollar el producto en un proceso de fases, que están ya establecidas para dicho desarrollo que son: Planeación, Requerimientos, Diseño, Implementación, Pruebas.

9.1.1.5 Diseño conceptual. Se va a implementar una plataforma web por protocolo CLIENTE-SERVIDOR en capas que facilite la construcción de programas pequeños, esta aplicación permitirá que los usuarios puedan construir algoritmos compuestos de primitivas básicas de programación, como los son: variables,

condicionales, ciclos, asignaciones entre otros, en este entorno se podrá compilar e interpretar dichos algoritmos.

Esta plataforma web permitirá:

- Crear, editar y eliminar programas pequeños que utilizan primitivas de programación.
- Compilar estos programas pequeños.
- Se tendrá un teclado personalizado dependiendo lo que el usuario este haciendo, este teclado le dará varias opciones de lo que el usuario puede hacer.
- Tendrá un login de acceso que funcionará con su email de preferencia.
- Consultas de todos los programas que se hayan realizado filtrándolo por su nombre respectivamente.
- Podrá proteger dichos programas en cualquier momento y estar disponibles en la nube cuando el usuario lo requiera.

9.1.1.6 Módulos

- Modulo seguridad. En este módulo se encargará de la autenticación, registro, recordar contraseña de los usuarios para que puedan acceder a la aplicación.
- Modulo compilador. En este módulo se encargará de compilar el código del usuario e interpretarlo con las funciones que se le permite al usuario desde el teclado.
- Modulo teclado dinámico. Este módulo se encarga de configurar el teclado dependiendo de la opción u opciones que el usuario este modificando.

9.1.1.7 Estimación preliminar del proyecto. Para la estimar el tamaño del producto se va a tener varios aspectos como el lenguaje de programación, también la experiencia en lenguajes de programación que ya conocemos para la realización del producto, sin extendernos en investigación de nuevos IDEs que comprometen al desarrollo del producto en tiempo.

Preliminar mente se va usar HTML 5 por parte de la interfaz gráfica, la parte de servicios en la nube se utilizará Web API MVC de Microsoft (Restfull), utilizando metodología de transferencia de datos como lo es JSON, para un óptimo manejo de datos por parte de la aplicación, en cuanto a la lógica de la aplicación usaremos un Framework llamado Angular JS desarrollado por Google, ya que utiliza un patrón MVVM que funciona asíncrono con HTML en cuanto al manejo del DOM, para la integridad de los datos se utilizara motor de base de datos SQL SERVER (Base de datos relacional), Además utilizaremos para la adaptabilidad de las diferentes pantallas usaremos un Framework llamado CSS desarrollo por Twitter que ayuda al comportamiento visual de la aplicación.

Para la estimación de tiempo de desarrollo del producto no tenemos una estimación acertada ya que por ser un compilador requiere un alto nivel de complejidad, y de análisis en cuanto al manejo por su estructura recursiva e infinita a la hora de hacer anidamiento de sentencias como ciclos o condicionales infinitos, pero aproximadamente se pretende como seis meses inicialmente.

9.1.2 Fase de requerimientos

9.1.2.1 Introducción. En esta fase realizaremos toda la documentación de todos los requerimientos funcionales y no funcionales con sus escenarios de calidad, para que el producto salga con todas las funcionalidades contempladas, para que el producto no salga sin funcionalidades que no se planearon con anterioridad.

9.1.2.2 Objetivo

- Realizar una buena especificación de requerimientos del producto.

- Implementar la especificación de todos los requerimientos para que el desarrollo de la aplicación sea la óptima posible.

9.1.2.3 Casos de uso. En este especificaremos los casos de uso ya que estas son las funcionalidades básicas por desarrollar para las funcionalidades del producto.

- Crear programa
- Guardar programa
- Modificar programa
- Eliminar programa
- Crear variable, ciclo y condicional
- Modificar variable, ciclo y condicional
- Eliminar variable, ciclo y condicional
- Ejecutar código escrito
- Iniciar sesión usuario

- Cerrar sesión usuario
- Crear una cuenta de usuario
- Cambiar Contraseña Usuario

Especificación de cada caso de uso:

Tabla 2. CU01: Crear programa

Numero de requerimiento	CU01	
Nombre de requerimiento	Acceso a la aplicación para crear un programa.	
Descripción del requerimiento	Funcionalidad para crear un programa.	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante selecciona la opción “NUEVO”.	6. La app despliega un modal para asignarle un nombre al programa.
	7. El estudiante ingresa el nombre del programa y presiona “Guardar”.	8. La app muestra un mensaje de confirmación para crear el nuevo programa.
	9. El estudiante presiona “OK”	10. La app muestra un mensaje de “Transacción Exitosa” indicando que se ha creado satisfactoriamente el programa con el nombre anteriormente asignado.
	11. El estudiante presiona “OK”	12. La app lo redirige a la pestaña de “Mis programas” mostrando el programa recién creado.
	13. El estudiante puede entrar a la aplicación indeterminadas veces y crear otros programas	

Autores

Tabla 3. CU02: Guardar programa

Numero de requerimiento	CU02	
Nombre de requerimiento	Acceso a la aplicación para guardar un programa.	
Descripción del requerimiento	Funcionalidad para guardar un programa.	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante selecciona la opción “NUEVO”.	6. La app despliega un modal para asignarle un nombre al programa.
	7. El estudiante ingresa el nombre del programa y presiona “Guardar”.	8. La app muestra un mensaje de confirmación para crear el nuevo programa.
	9. El estudiante presiona “OK”	10. La app muestra un mensaje de “Transacción Exitosa” indicando que se ha creado satisfactoriamente el programa con el nombre anteriormente asignado.
	11. El estudiante presiona “OK”	12. La app lo redirige a la pestaña de “Mis programas”.
	13. El estudiante selecciona el programa recién creado.	14. La app abre el editor de código.
	15. El estudiante puede modificar el editor de código y una vez termine presionará “PROTEGER” para guardar.	16. La app guarda y muestra un mensaje de “Transacción Exitosa” indicado que se guardó correctamente.
	17. El estudiante puede entrar a la aplicación indeterminadas veces y guardar otros programas.	

Autores

Tabla 4. CU03: Modificar programa

Numero de requerimiento	CU03	
Nombre de requerimiento	Acceso a la aplicación para modificar un programa.	
Descripción del requerimiento	Funcionalidad para modificar un programa.	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante selecciona uno de los programas anteriormente creados.	6. La app abre el editor de código.
	7. El estudiante puede modificar el contenido que está en el editor de código y una vez termine presionará “PROTEGER” para guardar.	8. La app guarda y muestra un mensaje de “Transacción Exitosa” indicado que se guardó correctamente.
	9. El estudiante puede entrar a la aplicación indeterminadas veces y modificar otros programas.	

Autores

Tabla 5. CU04: Eliminar programa

Numero de requerimiento	CU04	
Nombre de requerimiento	Acceso a la aplicación para eliminar un programa.	
Descripción del requerimiento	Funcionalidad para eliminar un programa.	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la

		pestaña “Mis programas”.
	5. El estudiante selecciona uno de los programas anteriormente creados.	6. La app abre el editor de código.
	7. El estudiante puede presionar la opción “Eliminar” para eliminar el programa completo.	8. La app muestra un mensaje de confirmación “Desea eliminar el programa Nombre.java”.
	9. El estudiante selecciona “OK”.	10. la app elimina el programa y muestra un mensaje de “Transacción Exitosa” indicando que se ha eliminado satisfactoriamente el programa.
	11. El estudiante presiona “OK”.	10. la app lo redirige a la pestaña de “Mis programas”.
	9. El estudiante puede entrar a la aplicación indeterminadas veces y eliminar otros programas.	

Autores

Tabla 6. CU05: Crear variables, métodos, ciclos y condicionales

Numero de requerimiento	CU05	
Nombre de requerimiento	Acceso al editor de código para crea una variable, un ciclo y un condicional.	
Descripción del requerimiento	Funcionalidad para crear una variable, un ciclo y un condicional.	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante selecciona uno de los programas anteriormente creados.	6. La app abre el editor de código.

	7. El estudiante toca la pantalla dentro de la estructura principal del editor de código.	8. La app muestra un teclado virtual con las múltiples opciones de seleccionar una inserción arriba, abajo y dentro del bloque seleccionado.
	9. El estudiante selecciona una inserción abajo.	10. la app muestra los diferentes tipos de inserción tales como los ciclos, condicionales, métodos, variables, asignaciones y la opción de atrás.
	11. El estudiante puede seleccionar cualquier opción	12. la app plasma la opción seleccionada ya sea una variable, un ciclo o un condicional.
	13. El estudiante puede entrar a la aplicación indeterminadas veces y agregar otras variables, métodos, ciclos y condicionales.	

Autores

Tabla 7. CU06: Modificación de variables, métodos, ciclos y condicionales

Numero de requerimiento	CU06	
Nombre de requerimiento	Acceso al editor de código para modificar una variable, un ciclo y un condicional.	
Descripción del requerimiento	Funcionalidad para modificar una variable, un ciclo y un condicional.	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante selecciona uno de los programas anteriormente creados.	6. La app abre el editor de código.

	7. El estudiante toca la pantalla dentro de la estructura principal del editor de código.	8. La app muestra un teclado virtual con las múltiples opciones de seleccionar una inserción arriba, abajo y dentro del bloque seleccionado.
	9. El estudiante selecciona una inserción abajo.	10. la app muestra los diferentes tipos de inserción tales como los ciclos, condicionales, métodos, variables, asignaciones y la opción de atrás.
	11. El estudiante puede seleccionar cualquier opción	12. la app plasma la opción seleccionada ya sea una variable, un ciclo o un condicional.
	13. El estudiante toca sobre la estructura de la sentencia plasmada en el editor de código ya sea sobre el tipo de dato, nombre o valor.	14. la app muestra un teclado virtual con las opciones de tipos de datos que puede tomar dicha sentencia o con un teclado alfanumérico en caso del nombre o valor.
	15. El estudiante puede entrar a la aplicación indeterminadas veces y modificar otras variables, métodos, ciclos y condicionales.	

Autores

Tabla 8. CU07: Eliminación de variables, métodos, ciclos y condicionales

Numero de requerimiento	CU07	
Nombre de requerimiento	Acceso al editor de código para eliminar una variable, un ciclo y un condicional.	
Descripción del requerimiento	Funcionalidad para eliminar una variable, un ciclo y un condicional.	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.

	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante selecciona uno de los programas anteriormente creados.	6. La app abre el editor de código.
	7. El estudiante toca la pantalla dentro de la estructura principal del editor de código.	8. La app muestra un teclado virtual con las múltiples opciones de seleccionar una inserción arriba, abajo y dentro del bloque seleccionado.
	9. El estudiante selecciona una inserción abajo.	10. la app muestra los diferentes tipos de inserción tales como los ciclos, condicionales, métodos, variables, asignaciones y la opción de atrás.
	11. El estudiante puede seleccionar cualquier opción	12. la app plasma la opción seleccionada ya sea una variable, un ciclo o un condicional.
	13. El estudiante toca sobre la estructura de la sentencia plasmada en el editor de código ya sea sobre el tipo de dato, nombre o valor.	14. la app muestra un teclado virtual con las opciones de tipos de datos que puede tomar dicha sentencia o con un teclado alfanumérico en caso del nombre o valor, además de un icono de eliminación.
	15. El estudiante selecciona el icono de eliminación.	16. La app elimina la sentencia seleccionada, en caso de ser un ciclo o una condicional elimina todo el bloque incluyendo su estructura interior (otras sentencias).

	17. El estudiante puede entrar a la aplicación indeterminadas veces y eliminar otras variables, métodos, ciclos y condicionales.	

Autores

Tabla 9. CU08: Ejecutar código escrito

Numero de requerimiento	CU08	
Nombre de requerimiento	Acceso al editor de código para ejecutar el código escrito.	
Descripción del requerimiento	Funcionalidad para compilar el código escrito.	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante selecciona uno de los programas anteriormente creados.	6. La app abre el editor de código.
	7. El estudiante toca la pantalla dentro de la estructura principal del editor de código.	8. La app muestra un teclado virtual con las múltiples opciones de seleccionar una inserción arriba, abajo y dentro del bloque seleccionado.
	9. El estudiante selecciona una inserción abajo.	10. la app muestra los diferentes tipos de inserción tales como los ciclos, condicionales, métodos, variables, asignaciones y la opción de atrás.
	11. El estudiante puede seleccionar cualquier opción	12. la app plasma la opción seleccionada ya sea una variable, un ciclo o un condicional.

	13. El estudiante puede presionar “RUN”	14. La app muestra la “Consola” de resultados en una nueva instancia.
	15. El estudiante puede entrar a la aplicación indeterminadas veces y ejecutar otros códigos ya escritos.	

Autores

Tabla 10.CU09: Iniciar Sesión Usuario

Numero de requerimiento	CU09	
Nombre de requerimiento	Acceso a la aplicación para iniciar sesión usuario	
Descripción del requerimiento	Funcionalidad para iniciar sesión usuario	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante puede entrar a la aplicación indeterminadas veces y validarse ante la app para la gestión de programas	

Autores

Tabla 11.CU010: Cerrar Sesión Usuario

Numero de requerimiento	CU010	
Nombre de requerimiento	Acceso a la aplicación para cerrar sesión usuario	
Descripción del requerimiento	Funcionalidad para cerrar sesión usuario	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.

	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña “Mis programas”.
	5. El estudiante presiona la opción de “MENU”	6. La app despliega el “MENU” con diferentes opciones.
	7. El estudiante selecciona la opción de “SALIR” para cerrar sesión en la app.	8. La app cierra sesión y lo redirige a la ventana de autenticación de la app.
	9. El estudiante puede entrar a la aplicación indeterminadas veces y seleccionar salir para cerrar sesión.	

Autores

Tabla 12. CU011: Crear una cuenta de usuario

Numero de requerimiento	CU011	
Nombre de requerimiento	Acceso a la aplicación para crear una cuenta de usuario	
Descripción del requerimiento	Funcionalidad para crear una cuenta de usuario	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante selecciona “Solicitar una cuenta”.	4. La app muestra una pantalla con un formulario próximo a llenar con los datos del estudiante.
	5. El estudiante llena el formulario con sus respectivos datos y presiona “Enviar”.	6. La app muestra un mensaje de “Tu cuenta ha sido creada correctamente” y redirige al estudiante a la vista principal del Login.
	13. El estudiante puede entrar a la aplicación indeterminadas veces y crear otros cuentas de usuario.	

Autores

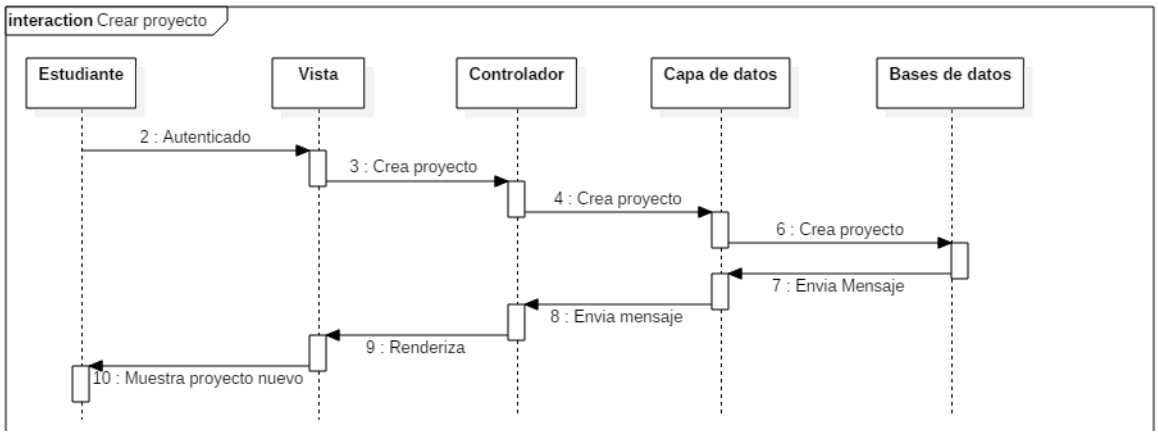
Tabla 13.CU012: Cambiar Contraseña Usuario

Numero de requerimiento	CU012	
Nombre de requerimiento	Acceso a la aplicación para cambiar clave de acceso de usuario	
Descripción del requerimiento	Funcionalidad para cambiar clave de acceso de usuario	
Secuencia	**Usuario**	**Sistema**
	1. El estudiante accede a la aplicación.	2. La app muestra como vista principal el Login.
	3. El estudiante se autentica.	4. La app valida los datos del estudiante y muestra como pantalla principal la pestaña "Mis programas".
	5. El estudiante presiona la opción de "MENU"	6. La app despliega el "MENU" con diferentes opciones.
	7. El estudiante selecciona "Mi cuenta"	8. La app muestra la información de la cuenta con opciones de modificación.
	9. El estudiante valida la contraseña anterior y las nuevas próximas a establecer y presiona "Cambiar".	10. La app muestra un mensaje de confirmación de cambio de clave.
	11. El estudiante confirma "OK".	12. La app muestra un mensaje de "Transacción Exitosa" indicando que se ha cambiado la clave satisfactoriamente.
	13. El estudiante puede entrar a la aplicación indeterminadas veces y cambiar la clave.	

Autores

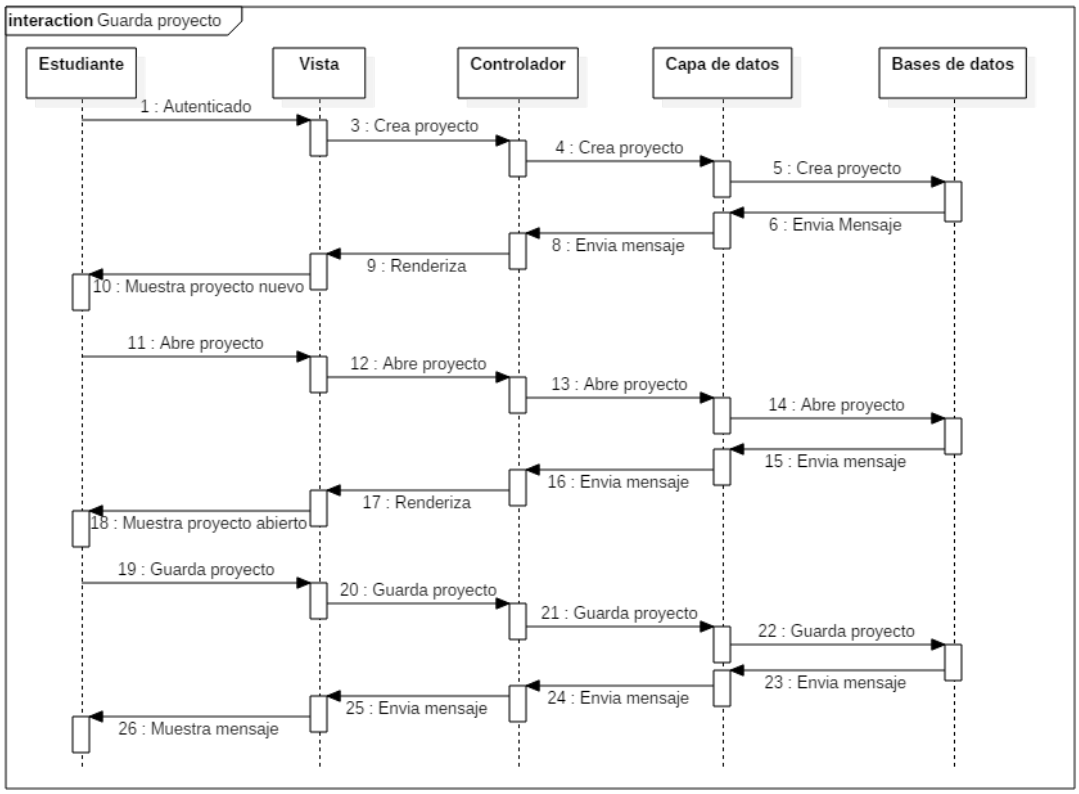
Diagrama de secuencias. Aquí mostraremos el flujo de eventos de cada uno de los casos de uso y de cómo interactúan entre sí:

Figura 3. Crear programa



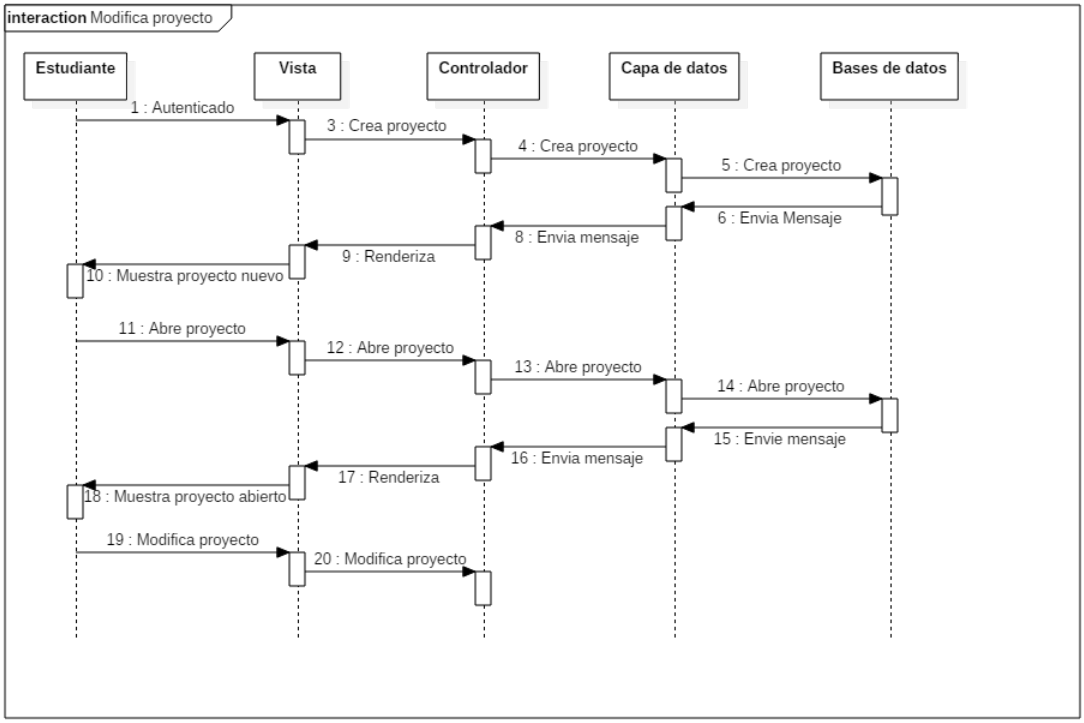
Autores

Figura 4. Guardar programa



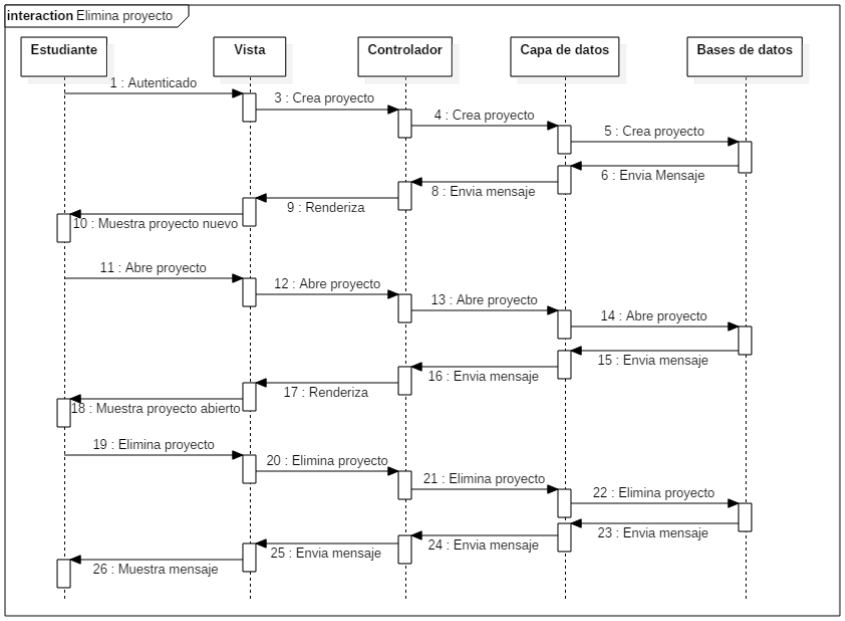
Autores

Figura 5. Modificar programa



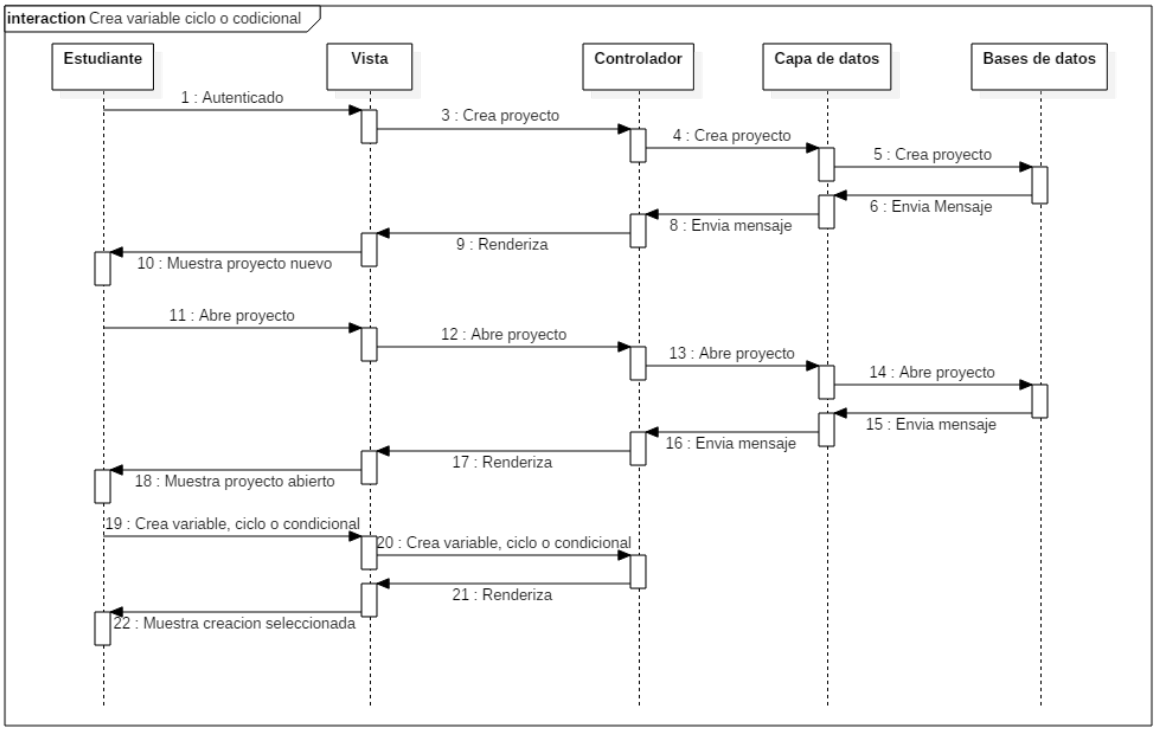
Autores

Figura 6. Eliminar programa



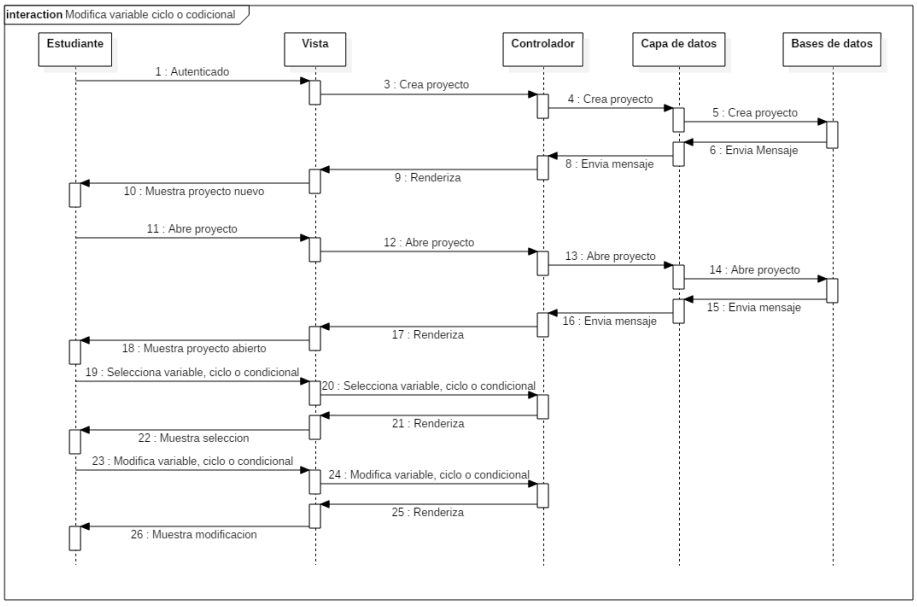
Autores

Figura 7. Crear variable, ciclo y condicional



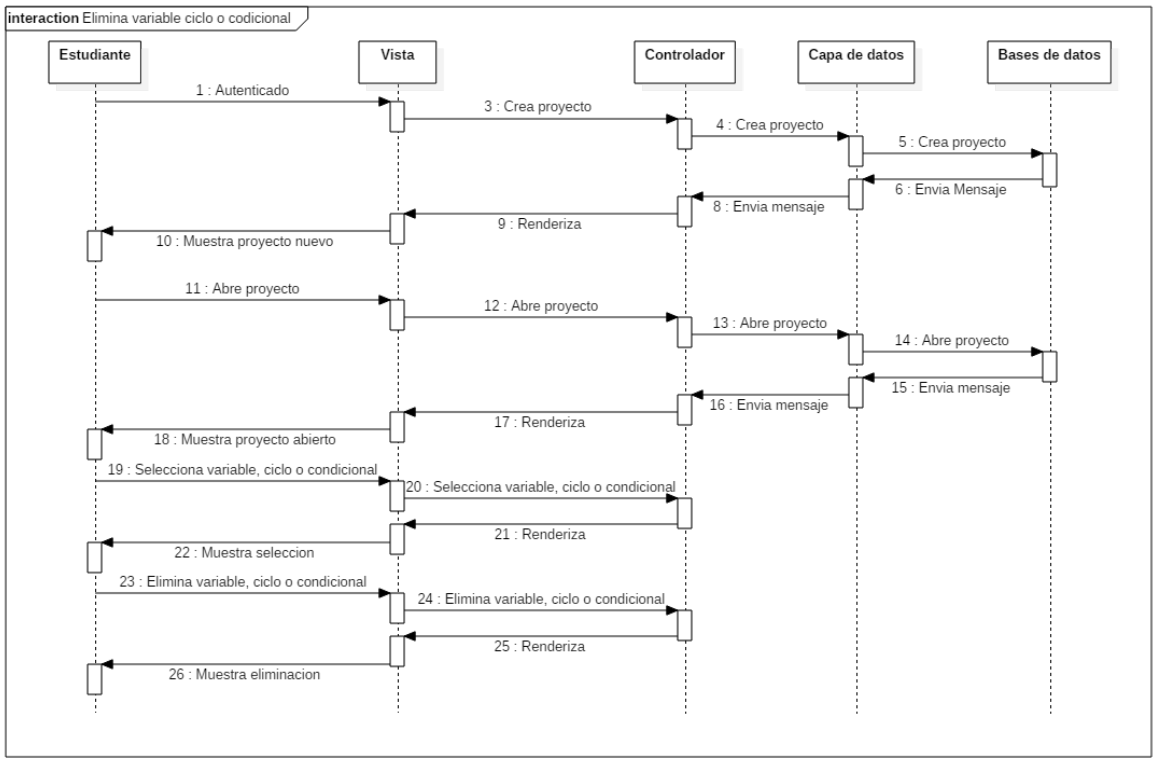
Autores

Figura 8. Modificar variable, ciclo y condicional



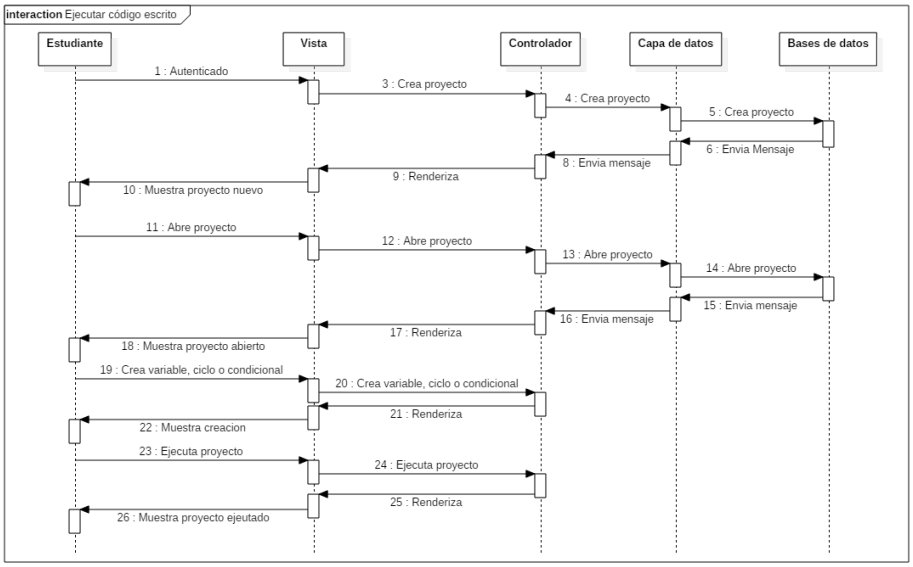
Autores

Figura 9. Eliminar variable, ciclo y condicional



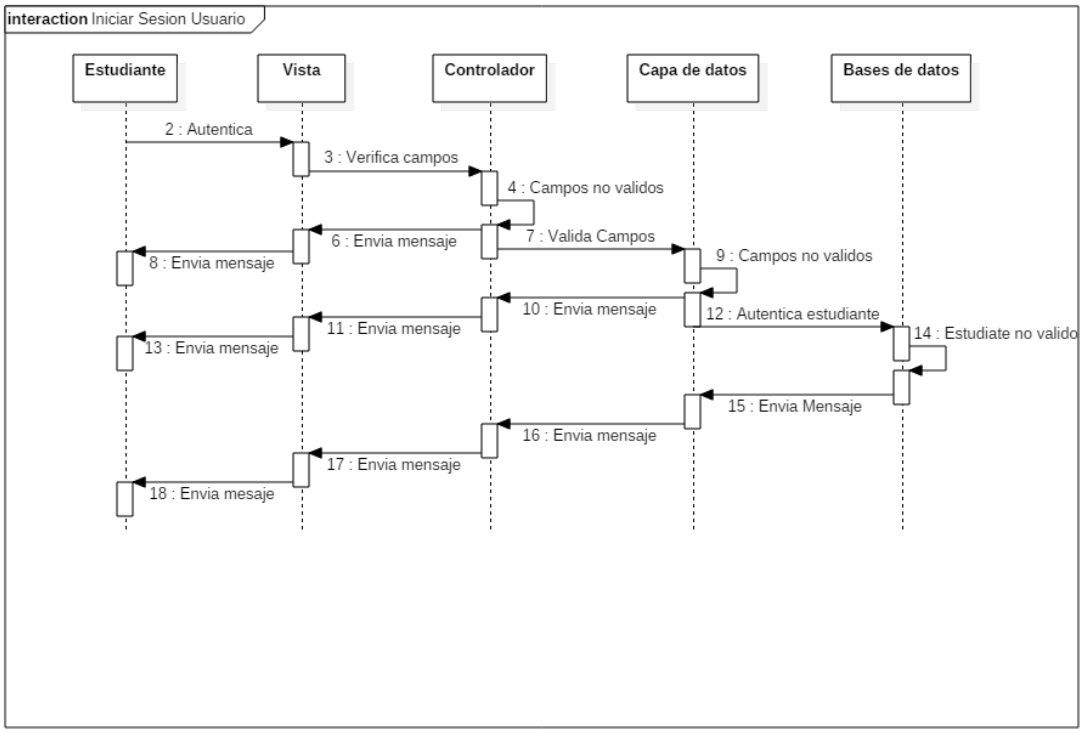
Autores

Figura 10. Ejecutar código escrito



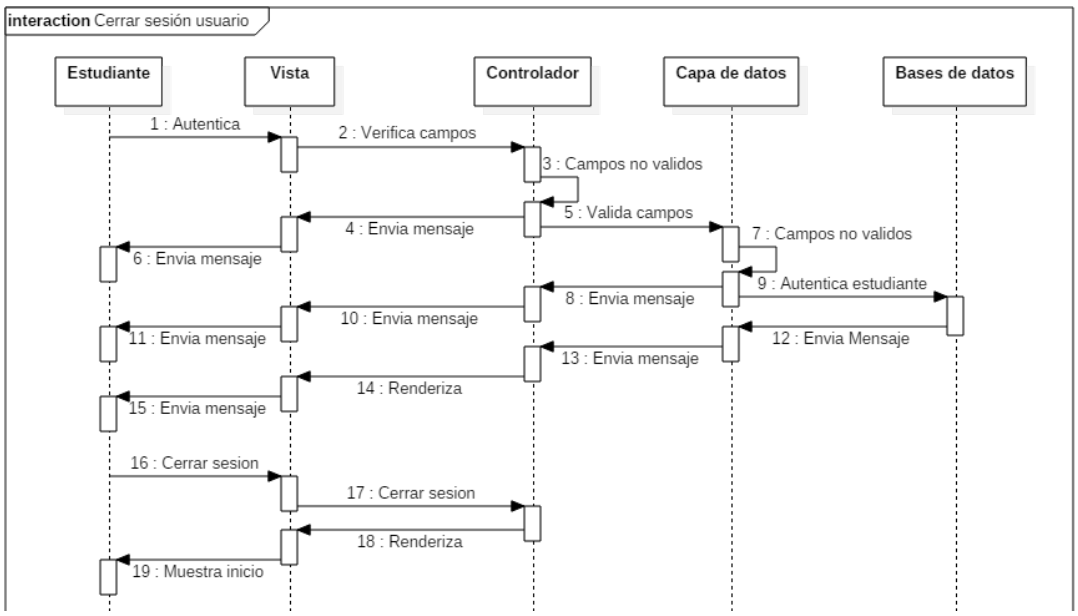
Autores

Figura 11. Iniciar sesión usuario



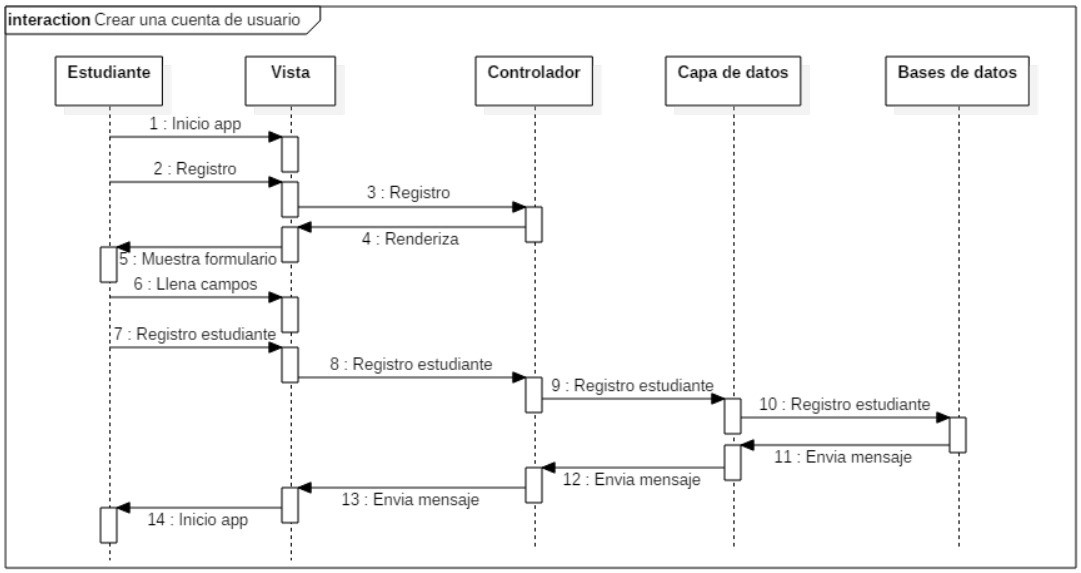
Autores

Figura 12. Cerrar sesión usuario



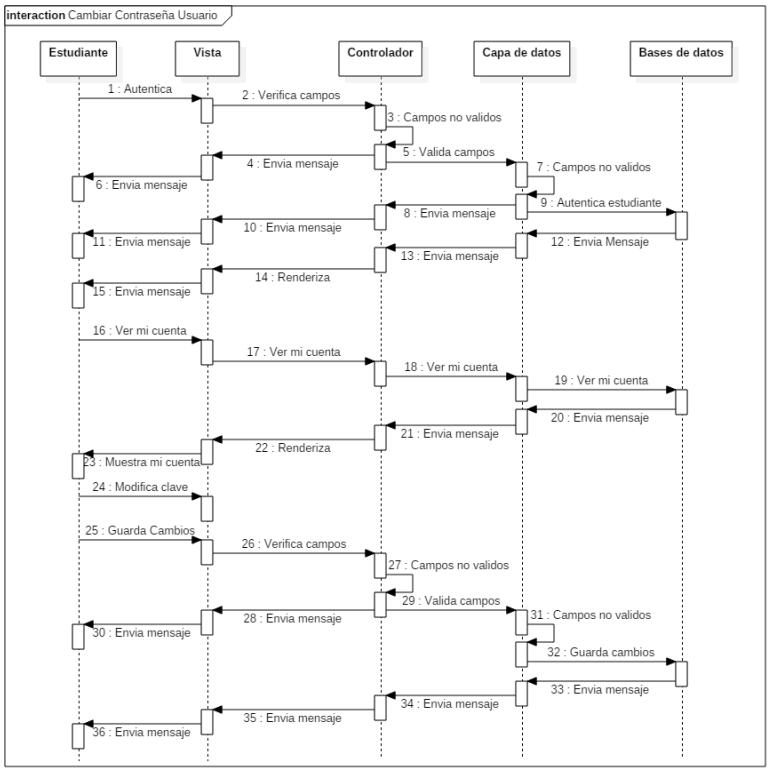
Autores

Figura 13. Crear una cuenta de usuario



Autores

Figura 14. Cambiar Contraseña Usuario



Autores

9.1.2.4 Requerimientos funcionales. Aquí contemplaremos las funcionalidades puntuales de que la aplicación puede hacer con un flujo de eventos y sus excepciones cuando se produzcan, estableciendo precondiciones o postcondiciones si lo requieren, entre los requerimientos funcionales encontramos:

- Gestionar programas (crear, editar, eliminar, ejecutar).
- Gestionar código (Seleccionador: En bloque, por partes, compilar, crear y asignar variables, ciclos, condicionales, operadores).
- Construir y evaluar expresiones aritméticas.
- Construir y evaluar expresiones lógicas.
- Construcción de la primitiva condicional if, else con anidamiento.
- Construcción de la primitiva cíclica for, con anidamiento.
- Consola de resultados.

Especificación de cada requerimiento:

Tabla 14. RQ01: Gestionar programas

Numero de requerimiento	RQ01
Nombre de requerimiento	Gestionar programas.
Prioridad	Alta
Critico	Si
Visible	Si
Resumen	Al entrar en la aplicación accederá al Login de la aplicación, el usuario se autenticará y entrará a gestionar los programas; crear, modificar, eliminar y ejecutar.
Entrada	
Salida	Programas creados, modificados, ejecutados.
Precondiciones	No haberse autenticado correctamente o no haber entrado a la pestaña de “Mis programas”.
Postcondiciones	Desde el inicio de la aplicación, la opción de gestión de programa está habilitada.
Excepciones	No todas las opciones de la plataforma permiten gestionar los programas.

Autores

Tabla 15. RQ02: Gestionar código

Numero de requerimiento	RQ02

Nombre de requerimiento	Gestionar código.
Prioridad	Alta
Critico	Si
Visible	Si
Resumen	Al autenticarse en la aplicación entrara a “Mis programas” y encontrara la opción de crear un programa nuevo, donde a media que se crean nuevas sentencias, estas se podrán seleccionar ya sea en bloque o por partes en el caso de una sentencia simple, cíclica o condicional, ya sea para eliminar toda la sentencia seleccionada o modificarla, donde por último se ejecuta.
Entrada	
Salida	Programas creados o editados pueden ser compilados y ejecutados.
Precondiciones	No haber entrado a la pestaña de “Mis programas” o no haber escrito por lo menos una línea de código en el editor de texto de la aplicación.
Postcondiciones	Desde la primera sentencia escrita, la gestión de código está habilitada, como crear, asignar, modificar y ejecutar.
Excepciones	No todas las opciones de la plataforma permiten gestionar código.

Autores

Tabla 16. RQ03: Construir y evaluar expresiones aritméticas

Numero de requerimiento	RQ03
Nombre de requerimiento	Construir y evaluar expresiones aritméticas.
Prioridad	Alta
Critico	Si
Visible	Si
Resumen	Al autenticarse en la aplicación entrara a “Mis programas” podrá seleccionar un programa o crear uno nuevo, al entrar en el editor de código podrá seleccionar tocando la pantalla dentro del método principal “main”, este toque habilitara un teclado virtual con la opción de agregar variables, ciclos, condicionales, una vez agregada la variable se le podrá asignar un nombre y un valor el cual puede ser una expresión aritmética.
Entrada	
Salida	Programas nuevos o modificados pueden construir y evaluar expresiones aritméticas.

Precondiciones	No haber entrado a la pestaña de “Mis programas” o no haber escrito por lo menos una línea de código en el editor de texto de la aplicación.
Postcondiciones	Desde la primera sentencia escrita, la construcción y evaluación de expresiones aritméticas está habilitada, como crear, asignar, modificar y ejecutar.
Excepciones	No todas las opciones de la plataforma permiten la construcción y evaluación de expresiones aritméticas.

Autores

Tabla 17. RQ04: Construir y evaluar expresiones lógicas

Numero de requerimiento	RQ04
Nombre de requerimiento	Construir y evaluar expresiones lógicas.
Prioridad	Alta
Critico	Si
Visible	Si
Resumen	Al autenticarse en la aplicación entrara a “Mis programas” podrá seleccionar un programa o crear uno nuevo, al entrar en el editor de código podrá seleccionar tocando la pantalla dentro del método principal “main”, este toque habilitara un teclado virtual con la opción de agregar variables, ciclos, condicionales, una vez agregado el ciclo o el condicional se le podrá asignar dentro de la estructura condicional una expresión lógica.
Entrada	
Salida	Programas nuevos o modificados pueden construir y evaluar expresiones lógicas.
Precondiciones	No haber entrado a la pestaña de “Mis programas” o no haber escrito por lo menos una línea de código en el editor de texto de la aplicación.
Postcondiciones	Desde la primera sentencia escrita, la construcción y evaluación de expresiones lógicas está habilitada, como crear, asignar, modificar y ejecutar.
Excepciones	No todas las opciones de la plataforma permiten la construcción y evaluación de expresiones lógicas.

Autores

Tabla 18. RQ05: Construcción de la primitiva condicional if, else con anidamiento

Numero de requerimiento	RQ05

Nombre de requerimiento	Construcción de la primitiva condicional if, else con anidamiento.
Prioridad	Alta
Critico	Si
Visible	Si
Resumen	Al autenticarse en la aplicación entrara a “Mis programas” podrá seleccionar un programa o crear uno nuevo, al entrar en el editor de código podrá seleccionar tocando la pantalla dentro del método principal “main”, este toque habilitara un teclado virtual con la opción de agregar variables, ciclos, condicionales, una vez agregado el condicional se le podrá agregar dentro de la estructura del cuerpo una nueva sentencia condicional, tocando la pantalla y seleccionando la condición próxima a agregar teniendo las mismas opciones de modificación y eliminación, quedando de esta manera anidadas la sentencias condicionales.
Entrada	
Salida	Programas nuevos o modificados pueden construir la primitiva condicional if, else con anidamiento.
Precondiciones	No haber entrado a la pestaña de “Mis programas” o no haber escrito por lo menos una línea de código en el editor de texto de la aplicación.
Postcondiciones	Desde la primera sentencia escrita, la construcción de la primitiva condicional if, else con anidamiento está habilitada, como crear, asignar, modificar y ejecutar.
Excepciones	No todas las opciones de la plataforma permiten la construcción de la primitiva condicional if, else con anidamiento.

Autores

Tabla 19. RQ06: Construcción de la primitiva cíclica for, con anidamiento

Numero de requerimiento	RQ06
Nombre de requerimiento	Construcción de la primitiva cíclica for, con anidamiento.
Prioridad	Alta
Critico	Si
Visible	Si
Resumen	Al autenticarse en la aplicación entrara a “Mis programas” podrá seleccionar un programa o crear uno nuevo, al entrar en el editor de código podrá seleccionar tocando la pantalla dentro del método principal “main”, este toque habilitara un teclado virtual con la opción de agregar variables, ciclos,

	condicionales, una vez agregado el ciclo se le podrá agregar dentro de la estructura del cuerpo un nuevo ciclo, tocando la pantalla y seleccionando el ciclo próximo a agregar teniendo las mismas opciones de modificación y eliminación, quedando de esta manera anidados los ciclos.
Entrada	
Salida	Programas nuevos o modificados pueden construir la primitiva cíclica for, con anidamiento.
Precondiciones	No haber entrado a la pestaña de “Mis programas” o no haber escrito por lo menos una línea de código en el editor de texto de la aplicación.
Postcondiciones	Desde la primera sentencia escrita, la construcción de la primitiva cíclica for, con anidamiento está habilitada, como crear, asignar, modificar y ejecutar.
Excepciones	No todas las opciones de la plataforma permiten la construcción de la primitiva cíclica for, con anidamiento.

Autores

Tabla 20. RQ07: Consola de resultados

Numero de requerimiento	RQ07
Nombre de requerimiento	Consola de resultados.
Prioridad	Alta
Critico	Si
Visible	Si
Resumen	Una vez autenticado el estudiante este podrá seleccionar uno de los programas que ya están creados en la pestaña de “Mis Programas” o crear uno nuevo, sobre el editor de código se encontrará un botón llamado “Run” que dará inicio a la ejecución del código escrito, mostrando los resultados en una nueva ventana denominada “Consola”.
Entrada	
Salida	Programas nuevos o modificados pueden ser ejecutados.
Precondiciones	No haber escrito por lo menos una línea o sentencia en el editor de código.
Postcondiciones	Desde la primera sentencia escrita, la consola estará habilitada para mostrar resultados una vez se haya ejecutado el código.
Excepciones	No todas las opciones de la plataforma permiten ver la consola de resultados.

Autores

9.1.2.5 Requerimientos no funcionales. En estos requerimientos encontramos aquellas características que son indispensables para el funcionamiento general de la aplicación entre las cuales encontramos:

- Usabilidad.
- Distribución geográfica.
- Persistencia
- Seguridad

Especificación de los requerimientos no funcionales:

Tabla 21. RQN01: Usabilidad

Numero de requerimiento	RQN01
Nombre	Usabilidad
Tipo	Indispensable
Prioridad	Esencial
Descripción: Que la aplicación se pueda ejecutar en diversas plataformas como lo son móviles y de escritorio.	
Criterios de aceptación: Que el usuario que no cuente con dispositivo de escritorio o móvil pueda acceder a la aplicación sin ninguna limitante.	

Autores

Tabla 22. RQN02: Distribución geográfica

Numero de requerimiento	RQN02
Nombre	Distribución geográfica
Tipo	Indispensable
Prioridad	Esencial
Descripción: Los usuarios del sistema pueden estar geográficamente distribuidos, lo que influye que el usuario puede estar en cualquier parte y acceder a la aplicación.	
Criterios de aceptación: Usuarios dispersos geográficamente tienen acceso al sistema.	

Autores

Tabla 23. RQN03: Persistencia

Numero de requerimiento	RQN03
Nombre	Persistencia
Tipo	Indispensable
Prioridad	Esencial
Descripción: Los datos de los programas deben persistir en el tiempo.	
Criterios de aceptación: Verificar que la existencia de la información del código del programa se encuentre registrada en la base de datos y que el usuario puedan consultarla desde sus diferentes dispositivos.	

Autores

Tabla 24. RQN04: Seguridad

Numero de requerimiento	RQN04
Nombre	Seguridad
Tipo	Indispensable
Prioridad	Esencial
Descripción: El sistema debe contar con un sistema de login con un email y contraseña y lo pueda utilizar para el ingreso de la aplicación.	
Criterios de aceptación: Debe conceder el acceso a cualquier usuario registrado en el sistema.	

Autores

9.1.2.6 Escenarios de calidad

- Escenario de usabilidad
  - Fuente
    - Usuario
  - Estimulo
    - Construir un programa desde un dispositivo móvil.
  - Ambiente
    - Ejecución bajo condiciones normales.
    - Ingresa desde Bogotá por medio de su dispositivo móvil.

- Respuesta
    - Se construye y compila un programa.

- Medida Respuesta
    - 100 % de los intentos se puede construir un programa desde la aplicación móvil.

- Escenario de persistencia
    - Fuente
        - Usuario

    - Estimulo
        - Agregar un programa para que persista en el tiempo y así poder ser visto desde otro dispositivo o lugar.

    - Ambiente
        - Ejecución bajo condiciones normales
        - Usuario trabajando desde su móvil y después en su computadora o Tablet.

    - Respuesta
        - Se agrega un programa y se consulta desde otro dispositivo.

    - Medida Respuesta
        - 100 % de los intentos para agregar el programa se cumplen satisfactoriamente.

- Escenario de distribución geográfica
    - Fuente
        - Usuario

    - Estimulo
        - Agregar un programa o registrar un usuario

    - Ambiente
        - Ejecución bajo condiciones normales
        - Usuario situado en Bogotá

    - Respuesta
        - Se agrega un usuario, un programa.

    - Medida Respuesta
        - 100 % de los intentos para agregar se cumplen satisfactoriamente.

- Escenario de seguridad

  - Fuente
    - Usuario

  - Estimulo
    - Usuarios que accedan a la aplicación sean los registrados en la base de datos.

  - Ambiente
    - Ejecución bajo condiciones normales
    - Usuario que va a ingresar a la aplicación desde la ciudad de Bogotá

  - Respuesta
    - Se valida correctamente si el usuario está en registrado en la aplicación y se le da ingreso.

  - Medida Respuesta
    - 100 % de los intentos para acceder a la aplicación se cumplen satisfactoriamente.

9.1.3 Fase de diseño

9.1.3.1 Introducción. En esta fase se tratará de abordar toda la parte de implementación donde se hablará de la arquitectura, se definirá los prototipos de las interfaces con modelos ya definidos el modelo de la base de datos se dejará ya establecida.

9.1.3.2 Objetivo

- Elaborar y diseñar una arquitectura y la especificación del diseño del software (Producto)

9.1.3.3 Objetivos específicos

- Diseñar las partes del producto y sus interfaces.
- Diseñar las vistas de la arquitectura del producto.
- Diseñar los modelos de clases, interfaces y datos.
- Asignación de responsabilidades de componentes del sistema.

9.1.3.4 Estrategia. En esta fase se tratará de abordar la mayor parte del problema del sistema antes de la implementación, para no hacer cambios inesperados en el diseño y así no comprometer el desarrollo de la aplicación con lo ya establecido y regirse por el diseño ya preestablecido donde se implementarán tres módulos de seguridad, compilador, teclado dinámico que este se configurará y esta dependerá de la primitiva que este activa en el momento.

9.1.3.5 Mecanismos de inspección de software

- Se establecerán las pruebas de usabilidad del teclado primordialmente.
- Revisión de nombres de variables, métodos, clases, definidas en la estructura del diseño.

9.1.3.6 Convenciones y estándares de diseño

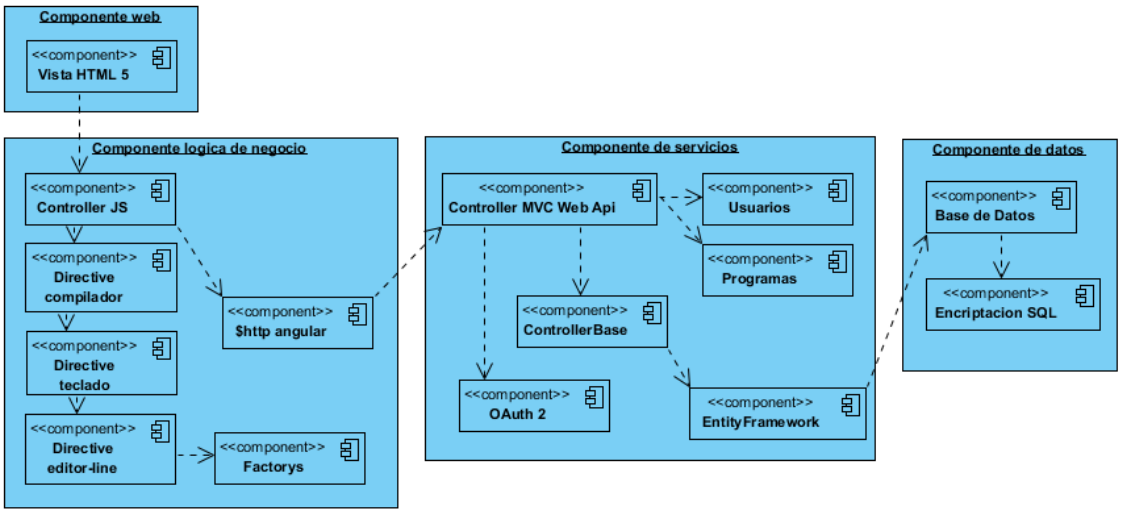
- Variables: Deben estar escritas en minúscula y los nombres deben ser claros.
- Clases: Primera letra en mayúscula y las demás en minúscula, definir nombres claros.
- Código General: Formateado, indentado para su fácil comprensión.

9.1.3.7 Arquitectura. La arquitectura que se implementara es de cuatro niveles o capas. Especialización de la arquitectura cliente-servidor donde la carga se divide en cuatro partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para la lógica de negocio (donde se encuentra la lógica del compilador), otra capa de servicios (intermedia para ofrecer servicios a diferentes dispositivos) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.

Para una especificación más detallada del sistema vamos a manejar varias vistas: Vistas de contexto, funcional, de información, de despliegue.

9.1.3.8 Vista funcional

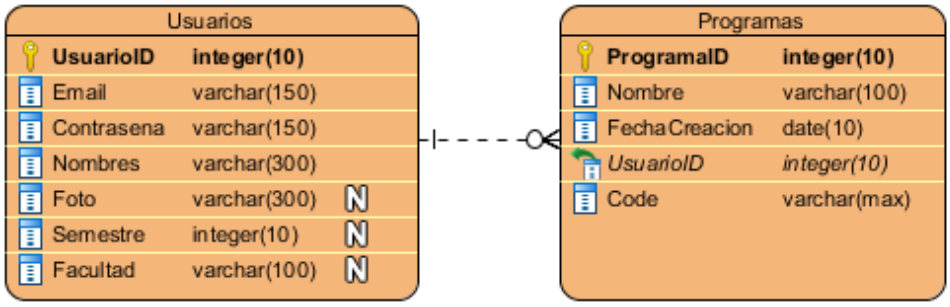
Figura 15. Vista funcional



Autores

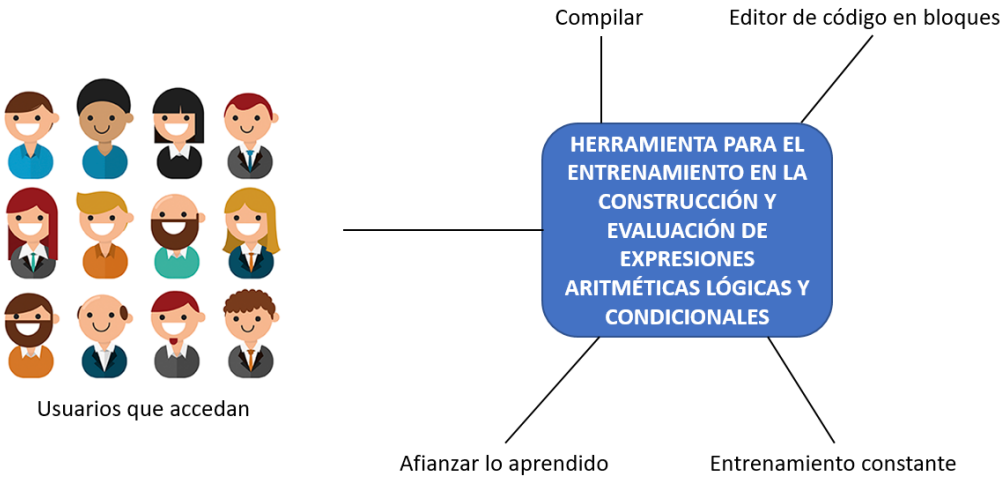
9.1.3.9 Vista de información o modelo de datos

Figura 16. Vista de información o modelo de datos



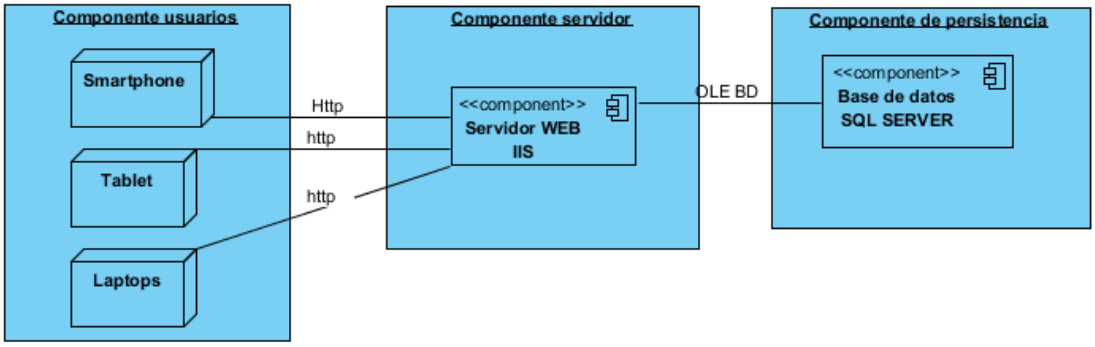
Autores

9.1.3.10 Vista de contexto
Figura 17. Vista de contexto



Autores

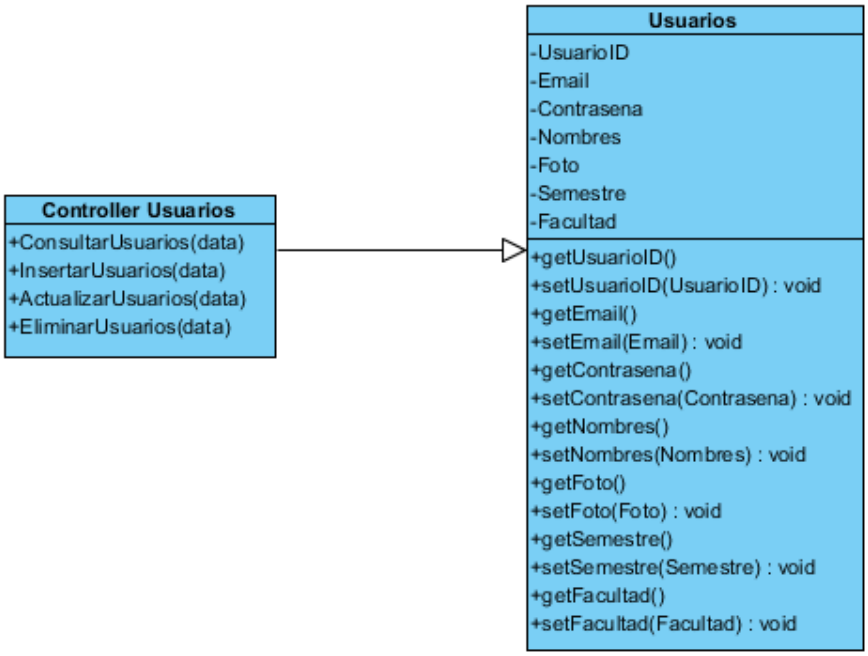
9.1.3.11 Vista de despliegue
Figura 18. Vista de despliegue



Autores

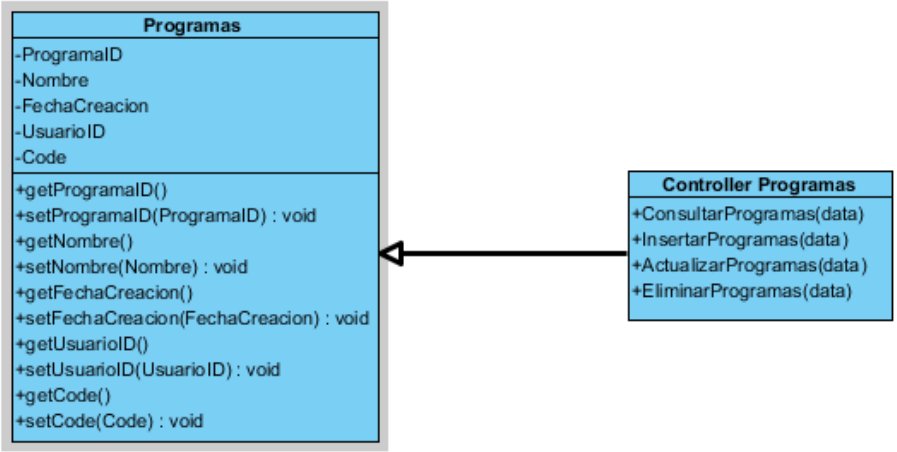
9.1.3.12 Diseño detallado. En el diseño detallado mostraremos toda la parte de los componentes y de las clases que tendrán el sistema, así como las interfaces de usuario que poseerá nuestro sistema y el modelo de la base de datos.

9.1.3.13 Modelo de clases por usuarios
Figura 19. Modelo de clases por usuarios



Autores

9.1.3.14 Modelo de clases por programas
Figura 20. Modelo de clases por programas



Autores

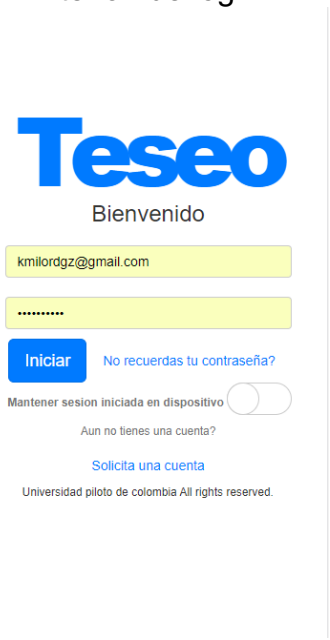
9.1.3.15 Responsabilidad de los componentes del sistema

Figura 21. Responsabilidad de los componentes del sistema

Autores

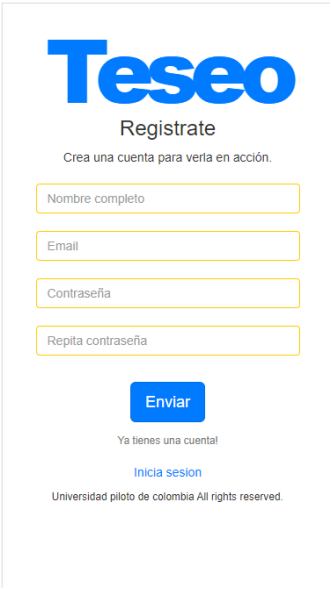
9.1.3.16 Modelos de interfaces “prototipos”. En esta sección hacemos un breve resumen de los prototipos de las interfaces para el desarrollo de la aplicación.

Figura 22. Interfaz de login



Autores

Figura 23. Registro de nuevos usuarios

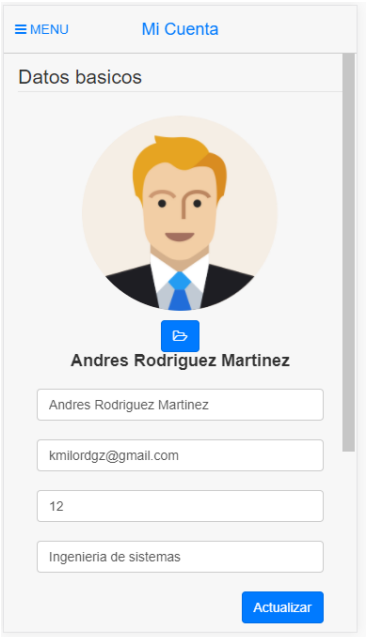


Autores

Figura 24. Recordar contraseña para usuarios

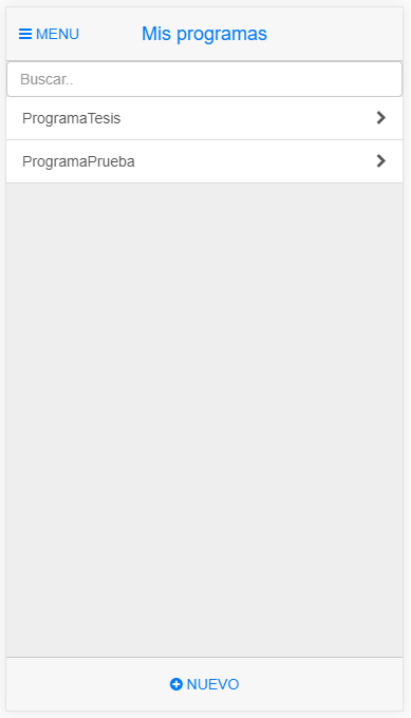
Autores

Figura 25. Mi cuenta



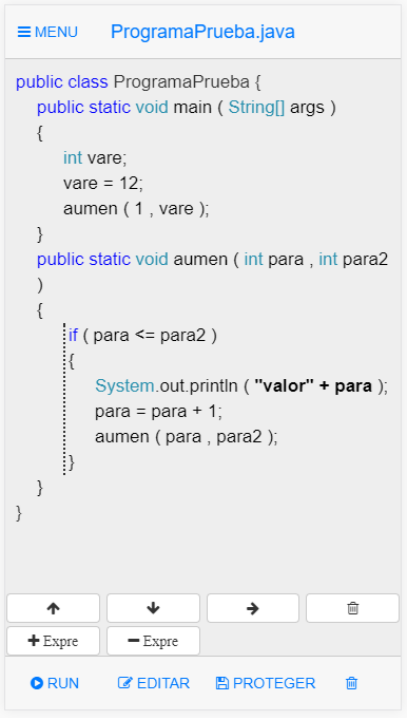
Autores

Figura 26. Listado de programas



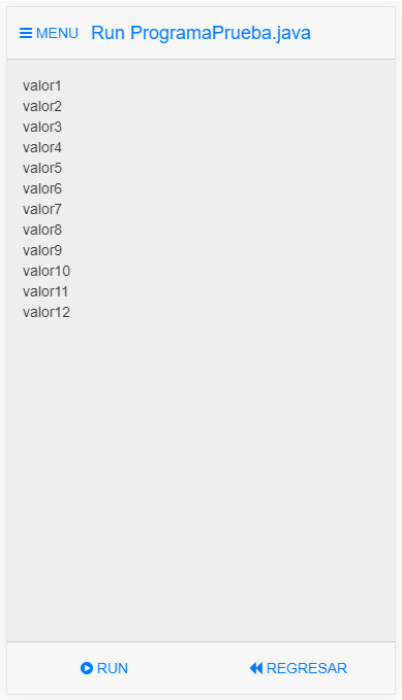
Autores

Figura 27. Editor de código en bloques



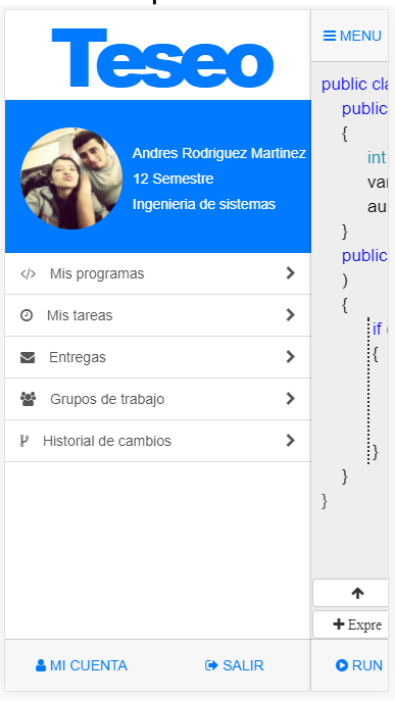
Autores

Figura 28. Consola



Autores

Figura 29. Vista previa del menú



Autores

9.1.3.17 Resolución de atributos de calidad. Para la resolución de los atributos de calidad ya establecidos se mencionará donde ya resolvimos estos atributos de calidad:

- Usabilidad. La usabilidad se va a contemplar ya que utilizaremos Framework CSS3 para la adaptabilidad en diferentes pantallas “Ver PROTOTIPOS DE INTERFAZ”

- Distribución geográfica. La distribución geográfica se va a implementar ya que el sistema cuenta con una plataforma web que el usuario podrá ver desde cualquier parte del planeta “Ver VISTA DESPLIEGUE o VISTA FUNCIONAL”.

- Persistencia. Para la persistencia se implementará la parte de bases de datos donde se guardarán la toda la información para tener la en tiempo real “Ver VISTA DE INFORMACION”

- Seguridad. La parte de seguridad se utilizará la verificación y validación de usuarios por medio de un inicio de sesión “Ver PROTOTIPOS DE INTERFAZ”.

9.1.4 Fase de implementación

9.1.4.1 Introducción. En esta fase trataremos la planeación de la implementación como del código y tareas diversas de acuerdo a lo que se va a desarrollar, para la elaboración de la implementación de la aplicación web para así optimizar tiempos de la misma implementación.

9.1.4.2 Objetivo. Elaborar y diseñar la fase de implementación para planear las tareas de implementación.

9.1.4.3 Criterios de entrada

- Estrategia del proyecto.

- Fase de requerimientos.

- Fase de diseño.

9.1.4.4 Estándares de Implementación

- Estándares de codificación, contenido y documentación. "La especificación del lenguaje C# no define un estándar de codificación. Sin embargo, Microsoft utiliza las instrucciones de este tema para desarrollar ejemplos y documentación.

Las convenciones de codificación tienen los objetivos siguientes:

- Crean una apariencia coherente en el código, para que los lectores puedan centrarse en el contenido, no en el diseño.
- Permiten a los lectores comprender el código más rápidamente al hacer suposiciones basadas en la experiencia anterior.
- Facilitan la copia, el cambio y el mantenimiento del código.
- Muestran los procedimientos recomendados de C#."[20]

Para la elaboración de nuestro producto utilizaremos la nomenclatura y convenciones dadas por Microsoft para Visual Studio que se encuentra relacionada en la documentación en su sitio web.

- Estrategias para inspección y revisión de código. Las inspección y revisión del código son importantes a la hora de identificar errores, mal funcionamiento o anomalías en el código, por ello es importante la revisión del mismo y por eso para nuestro producto se hizo una revisión individual por todo el grupo del proyecto, para así discutir errores encontrados y como solucionarlos, así partiendo de una retroalimentación del grupo para así mitigar los errores encontrados en la aplicación.

Además, se evidencio que el propio programador a la hora de realizar estas pruebas, nos dimos cuenta de que los errores de su razonamiento fueron los que originaron los errores originales en el código fuente de la aplicación, por ello recurrimos a persona ajenas al proceso como los asesores del proyecto para hacer un mejor testeo de la aplicación.

[20] MICROSOFT, Convenciones de código de C# (Guía de programación de C#) [En línea], Microsoft, [citado el 1 de noviembre de 2017], disponible desde: https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/inside-a-program/coding-conventions.

- Estrategias para garantizar la calidad de los productos de software. La estrategia que utilizamos para la calidad del producto es una revisión de todo el código fuente, es la revisión por los integrantes del equipo y asesores para hacer una retroalimentación de los errores encontrados.

- Planeación de la Implementación. La implementación se realizó de los módulos mirando en el orden que se requería para avanzar en la aplicación, primero se realizó el módulo de seguridad, luego el módulo del compilador y por terminar el módulo del teclado con el que más dificultad presentaba, ya que por su nivel de complejidad se presentó más dificultad para desarrollar, hasta conseguir una versión estable.

- Inspección del diseño detallado. El grupo del proyecto realizo una inspección de la fase de diseño volviendo a revisar las especificaciones del desarrollo, para hacer retroalimentación de los errores encontrados.

- Codificación. La codificación se hizo en conjunto con las dos personas del grupo que tienen más experiencia en desarrollo de software, y se sometió a pruebas por todos los integrantes del grupo.

- Inspección de código. La inspección del código se realizó con cada uno de los miembros del equipo o grupo que reviso minuciosamente el código producido para encontrar errores en el código como estándares de código o de lógica.

9.1.5 Fase de pruebas de integración. En esta fase se documentará la información sobre las pruebas de integración hechas a la aplicación para la aceptación de las mismas y cumplir con los casos de uso documentados en la fase de requerimientos.

9.1.5.1 Objetivo

- Elaborar y diseñar las pruebas para integrar los componentes del producto, pruebas de integración, aceptación y carga.

9.1.5.2 Objetivos específicos

- Asegurar la calidad de los componentes antes de probarlos.

- Elaborar pruebas de integración y sistema.

9.1.5.3 Criterios de entrada

- Fase de lanzamiento y estrategia.
- Fase de requerimientos.
- Fase de diseño.

9.1.5.4 Especificación de pruebas de integración. En este ítem vamos a especificar las pruebas realizadas cumpliendo los casos de uso documentados en la fase de requerimientos, para la aceptación de cada uno de los mismos requerimientos relacionados.

Tabla 25. Caso de prueba crear un programa

CASO DE PRUEBA CREAR UN PROGRAMA	
Descripción	Caso básico funcionalidad para crear un programa.
Fecha realización	12/11/2017
Realizada por	Nelson Gabriel Cardenas
Objetivo de la prueba	Validar CU01. Acceso a la aplicación para crear un programa.
Pre-Requisitos	• Tener conexión con el API desde el aplicativo. • Contar con unas credenciales de acceso validas en la herramienta.

DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario da clic en la caja de texto de email	El sistema despliega el teclado para el ingreso del correo electrónico	Muestra
1.2	El usuario escribe la contraseña correspondiente	El sistema captura la contraseña ingresada por el usuario	No muestra nada
1.3	El usuario da clic en el botón de iniciar sesión	El sistema valida la información y si es correcta envía al usuario a un menú principal	Muestra menú
1.4	El usuario da clic en el botón de nuevo programa	El sistema despliega una ventana emergente que permite ingresar el nombre del programa nuevo	Muestra ventana emergente

1.5	El Usuario ingresa el nombre del programa	El sistema captura el nombre del programa nuevo	No muestra nada
1.6	El usuario da clic en el botón de guardar	El sistema llama el API de registro de programas nuevos, envía la información y recarga el listado de programas	Aprobado
OBSERVACIONES			
El resultado al validar toda la información tiene que ser exitoso.			

Autores

Tabla 26. Caso de prueba proteger cambios

CASO DE PRUEBA PROTEGER CAMBIOS	
Descripción	Caso básico de proteger cambios.
Fecha realización	12/11/2017
Realizada por	Nelson Gabriel Cardenas
Objetivo de la prueba	Validar CU01. Proteger cambios.
Pre-Requisitos	• Tener conexión con el API de programas. • Contar con un programa previamente creado • Estar autenticado en la herramienta

DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario selecciona un programa de la lista de programas creados	El sistema despliega la interfaz de edición de código	Muestra el editor de código
1.2	El usuario modifica el programa	El sistema captura los cambios sobre la estructura del programa	No muestra nada
1.3	El usuario da clic en el botón de proteger	El sistema muestra un mensaje de confirmación de la transacción.	Aprobado
OBSERVACIONES			
El resultado al validar toda la información tiene que arrojar una alerta.			

Autores

Tabla 27. Caso de prueba modificar programa

CASO DE PRUEBA MODIFICAR PROGRAMA

Descripción	Caso básico de modificar programa.
Fecha realización	12/11/2017
Realizada por	Nelson Gabriel Cardenas
Objetivo de la prueba	Validar CU03. Modificar programa
Pre-Requisitos	• Tener conexión con el API de programas. • Estar autenticado como usuario valido del sistema. • Contar con un programa previamente creado

DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario selecciona un programa de la lista de programas	El sistema despliega la interfaz de edición de código	Muestra el editor de código
1.2	El usuario da clic en el botón editar	El sistema despliega una ventana emergente que permite cambiar el nombre del programa	Muestra ventana emergente
1.3	El usuario modifica el nombre del programa	El sistema captura el nuevo nombre del programa	No muestra nada
1.4	El usuario da clic en el botón de guardar	El sistema envía la información al método de guardado del API de programas y muestra una alerta de transacción exitosa	Aprobada
OBSERVACIONES			
El resultado al validar toda la información tiene que arrojar una alerta.			

Autores

Tabla 28. Caso de prueba eliminar programa

CASO DE PRUEBA ELIMINAR PROGRAMA	
Descripción	Caso básico de eliminar un programa que se encuentra registrado en la base de datos.
Fecha realización	12/11/2017
Realizada por	Nelson Gabriel Cardenas

Objetivo de la prueba	Validar CU04. Eliminar programa.
Pre-Requisitos	• Tener conexión con el API de programas. • Contar con un programa previamente creado. • Estar autenticado como usuario valido del sistema.

DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario selecciona un programa de la lista de programas	El sistema despliega la interfaz de edición de código	Muestra el editor de código
1.2	El usuario da clic en el botón eliminar	El sistema despliega una ventana de confirmación.	Muestra ventana emergente
1.3	El usuario da clic en el botón de confirmar	El sistema enviar la petición de eliminar el programa de la base de datos y muestra una alerta de transacción exitosa	Aprobado
OBSERVACIONES			
El resultado al validar toda la información del usuario encontrado debe ser exitosa.			

Autores

Tabla 29. Caso de prueba crear variables, métodos, ciclos y condicionales

CASO DE PRUEBA CREAR VARIABLES, MÉTODOS, CICLOS Y CONDICIONALES	
Descripción	Acceso al editor de código para crea una variable, un ciclo y un condicional.
Fecha realización	12/11/2017
Realizada por	Nelson Gabriel Cardenas
Objetivo de la prueba	Validar CU05. crear variables, métodos, ciclos y condicionales.
Pre-Requisitos	• Tener conexión con el API de programas. • Contar con un programa previamente creado. • Ser y estar autenticado como usuario valido del sistema.

DETALLE DE PRUEBA

Paso #	Acción de usuario	Respuesta esperada del sistema	Resultados
1.1	El usuario selecciona un programa de la lista de programas	El sistema despliega la interfaz de edición de código	Muestra el editor de código
1.2	El usuario selecciona el método principal	El sistema muestra un teclado de funciones de bloque el cual permite crear objetos completos	Muestra teclado
1.3	El usuario da clic en la función requerida (método, variable, ciclo o condicional)	El sistema agrega el bloque de código dentro del método	Muestra el cambio sobre el editor
1.4	El usuario parametriza las opciones que le da cada uno de los objetos agregados	El sistema realiza el cambio sobre el objeto de código	Aprobado
OBSERVACIONES			
El resultado al agregar los objetos debe verse en el editor en línea.			

Autores

Tabla 30. Caso de prueba modificar variables, métodos, ciclos y condicionales

CASO DE PRUEBA MODIFICAR VARIABLES, MÉTODOS, CICLOS Y CONDICIONALES			
Descripción	Acceso al editor de código para modificar una variable, un ciclo y un condicional.		
Fecha realización	12/11/2017		
Realizada por	Nelson Gabriel Cardenas		
Objetivo de la prueba	Validar CU06 modificar una variable, un ciclo y un condicional.		
Pre-Requisitos	• Tener conexión con el API de programas. • Contar con un programa previamente creado. • Ser y estar autenticado como usuario valido del sistema.		
DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario selecciona un programa de la lista de programas	El sistema despliega la interfaz de edición de código	Muestra el editor de código

1.2	El usuario selecciona el bloque de código que desea modificar	El sistema muestra un teclado de funciones de bloque el cual permite realizar acciones sobre el bloque	Muestra teclado
1.4	El usuario parametriza las opciones que le da cada uno de los objetos agregados	El sistema realiza el cambio sobre el objeto de código	Aprobado
OBSERVACIONES			
El resultado al editar los objetos debe verse en el editor en línea			

Autores

Tabla 31. Caso de prueba eliminación de variables, métodos, ciclos y condicionales

CASO DE PRUEBA ELIMINACIÓN DE VARIABLES, MÉTODOS, CICLOS Y CONDICIONALES			
Descripción	Eliminación de variables, métodos, ciclos y condicionales.		
Fecha realización	12/11/2017		
Realizada por	Nelson Gabriel Cardenas		
Objetivo de la prueba	Validar CU07. Eliminación de variables, métodos, ciclos y condicionales.		
Pre-Requisitos	• Tener conexión con el API de programas. • Contar con un programa previamente creado. • Ser y estar autenticado como usuario valido del sistema		
DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario selecciona un programa de la lista de programas	El sistema despliega la interfaz de edición de código	Muestra el editor de código
1.2	El usuario selecciona el bloque de código que desea eliminar	El sistema muestra un teclado de funciones de bloque el cual permite realizar acciones sobre el bloque y muestra un botón con una caneca que permite eliminar el bloque	Muestra teclado
1.4	El usuario da clic sobre el botón de la caneca	El sistema elimina el bloque de código junto con todas las instrucciones que se encuentren dentro de este bloque	Aprobado
OBSERVACIONES			

El resultado al editar los objetos debe verse en el editor en línea

Autores

Tabla 32. Caso de prueba ejecutar código escrito

CASO DE PRUEBA EJECUTAR CODIGO ESCRITO			
Descripción	Funcionalidad para compilar el código escrito.		
Fecha realización	12/11/2017		
Realizada por	Nelson Gabriel Cardenas		
Objetivo de la prueba	Validar CU09. Ejecutar código escrito.		
Pre-Requisitos	• Tener conexión con el API de programas. • Contar con un programa previamente creado. • Ser y estar autenticado como usuario valido del sistema • Tener el programa escogido sin errores lógicos y sintácticos.		
DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario selecciona un programa de la lista de programas	El sistema despliega la interfaz de edición de código	Muestra el editor de código
1.2	El usuario da clic sobre el botón de ejecutar	El sistema compila el código y muestra la consola de resultados	Muestra consola de resultados
OBSERVACIONES			
El resultado al compilar el código debe ser la ejecución del código o los errores encontrados en la ejecución.			

Autores

Tabla 33. Caso de prueba iniciar sesión

CASO DE PRUEBA INICIAR SESIÓN	
Descripción	Funcionalidad para iniciar sesión.
Fecha realización	12/11/2017
Realizada por	Nelson Gabriel Cardenas

Objetivo de la prueba	Validar CU10. Iniciar Sesión.
Pre-Requisitos	• Tener conexión con el API de usuarios. • Estar registrado como usuario valido del sistema

DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario da clic en la caja de texto de email	El sistema despliega el teclado para el ingreso del correo electrónico	Muestra
1.2	El usuario escribe la contraseña correspondiente	El sistema captura la contraseña ingresada por el usuario	No muestra nada
1.3	El usuario da clic en el botón de iniciar sesión	El sistema valida la información y si es correcta envía al usuario a un menú principal	Aprobado

OBSERVACIONES
El resultado al validar toda la información del usuario debe arrojar el menú principal.

Autores

Tabla 34. Caso de prueba cerrar sesión

CASO DE PRUEBA CERRAR SESIÓN	
Descripción	Funcionalidad para cerrar sesión de usuario.
Fecha realización	12/11/2017
Realizada por	Nelson Gabriel Cardenas
Objetivo de la prueba	Validar CU011. Cerrar sesión.
Pre-Requisitos	• Tener conexión con el API de usuarios. • Ser y estar autenticado como usuario valido del sistema

DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario da clic en el botón de cerrar sesión	El sistema elimina los datos en sesión y redirige al formulario de inicio de sesión	Aprueba

OBSERVACIONES
El resultado al cerrar sesión debe mostrar el formulario de inicio de sesión.

Autores

Tabla 35. Caso de prueba crear una cuenta de usuario

CASO DE PRUEBA CREAR UNA CUENTA DE USUARIO			
Descripción	Funcionalidad para crear una cuenta de usuario		
Fecha realización	12/11/2017		
Realizada por	Nelson Gabriel Cardenas		
Objetivo de la prueba	Validar CU012 Crear una cuenta de usuario.		
Pre-Requisitos	• Tener conexión con el API de usuarios. • Estar registrado como usuario valido del sistema		
DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario da clic sobre el botón “Solicita una cuenta”	El sistema despliega la interfaz de registro de usuario nuevo	Muestra
1.2	El usuario digita el nombre	El sistema captura la información del formulario.	No muestra nada
1.3	El usuario digita el email	El sistema captura la información del formulario.	No muestra nada
1.4	El usuario digita la contraseña	El sistema captura la información del formulario.	No muestra nada
1.5	El usuario digita nuevamente la contraseña	El sistema captura la información del formulario.	No muestra nada
1.6	El usuario da clic en el botón enviar	El sistema envía la información al API de registro y muestra una alerta de confirmación de proceso exitoso	Muestra alerta
OBSERVACIONES			
El resultado al validar toda la información debe arrojar una alerta.			

Autores

Tabla 36. Caso de prueba cambiar contraseña usuario

CASO DE PRUEBA CAMBIAR CONTRASEÑA USUARIO	
Descripción	Funcionalidad para cambiar clave de acceso de usuario
Fecha realización	12/11/2017
Realizada por	Nelson Gabriel Cardenas

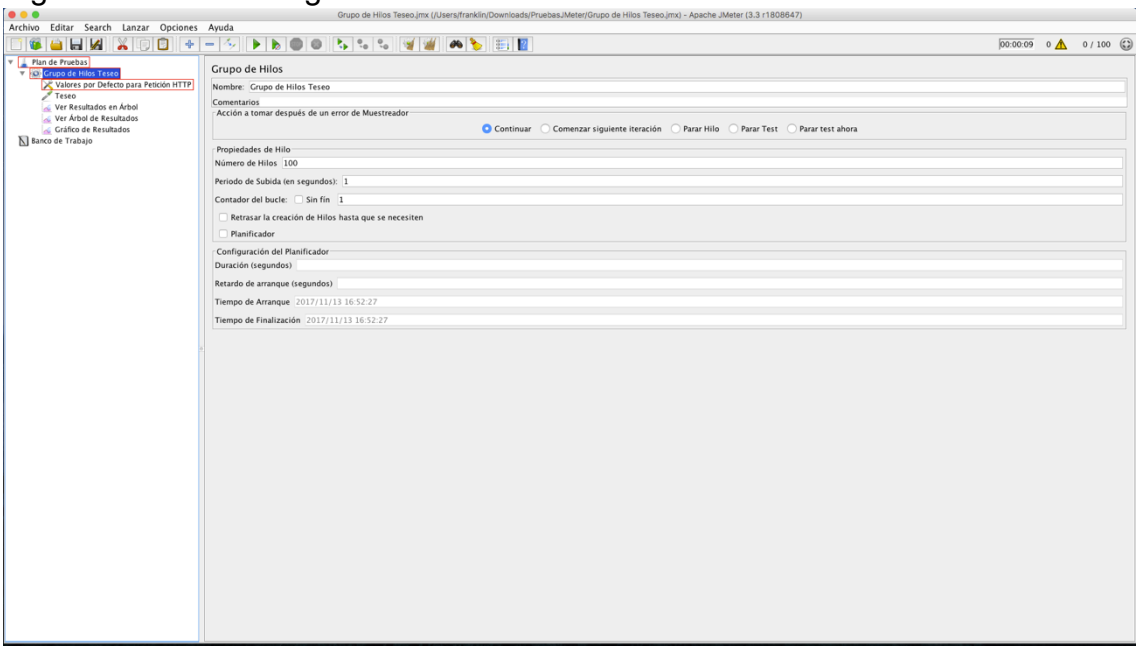
Objetivo de la prueba	Validar CU013. Cambiar Contraseña Usuario.		
Pre-Requisitos	• Tener conexión con la BD desde el aplicativo. • Ser y estar autenticado como usuario valido del sistema		
DETALLE DE PRUEBA			
Paso #	**Acción de usuario**	**Respuesta esperada del sistema**	**Resultados**
1.1	El usuario da clic en el menú de mi cuenta	El sistema despliega la interfaz de perfil de usuario.	Muestra
1.2	El usuario digita la clave anterior	El sistema captura la información del formulario	No muestra nada
1.3	El usuario digita la nueva clave	El sistema captura la información del formulario	No muestra nada
1.4	El usuario digita nuevamente la clave nueva	El sistema captura la información del formulario	No muestra nada
1.5	El usuario oprime el botón de guardar cambios	El sistema envía la información mediante el API de usuarios.	Aprobado
OBSERVACIONES			
El resultado al validar toda la información debe arrojar una alerta.			

Autores

9.1.5.5 Especificación de pruebas de carga. Esta prueba consiste en medir la carga, rendimiento y el estrés de la aplicación la cual mide la cantidad de usuarios que puede soportar y aguantar la aplicación sin generar error, para esto se hicieron 5 pruebas en la aplicación de JMeter que se mencionan a continuación:

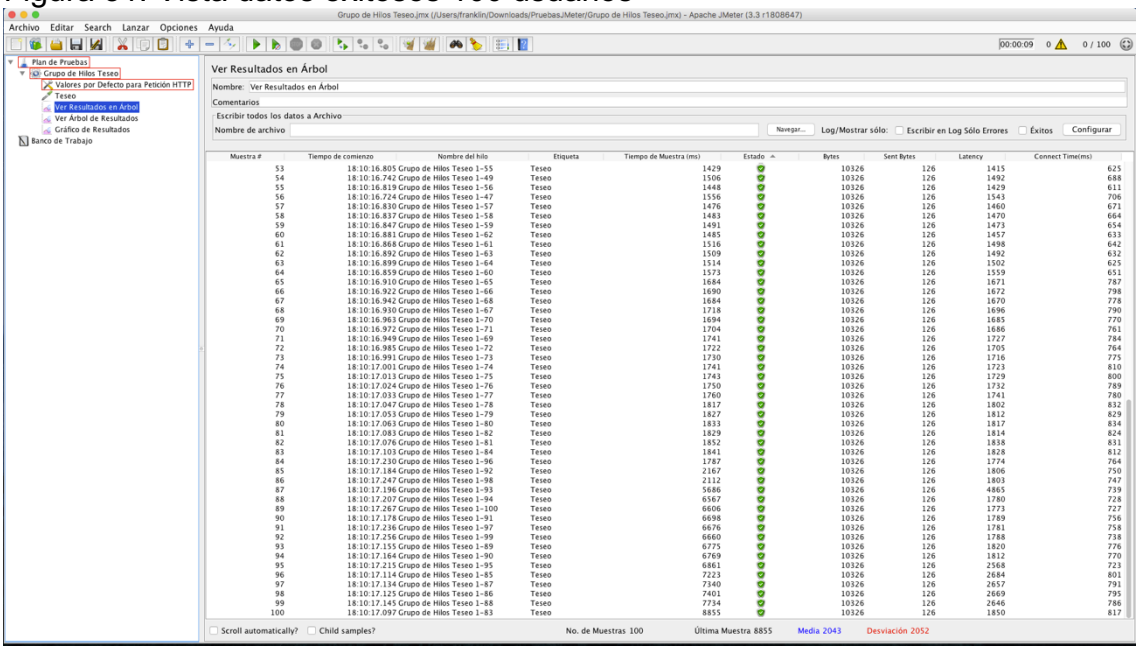
- Prueba 1 (100 Usuarios). Se realizo una prueba de que pueda responder a 100 usuarios sin errores, que se cumplió satisfactoriamente.

Figura 30. Vista configuración 100 usuarios



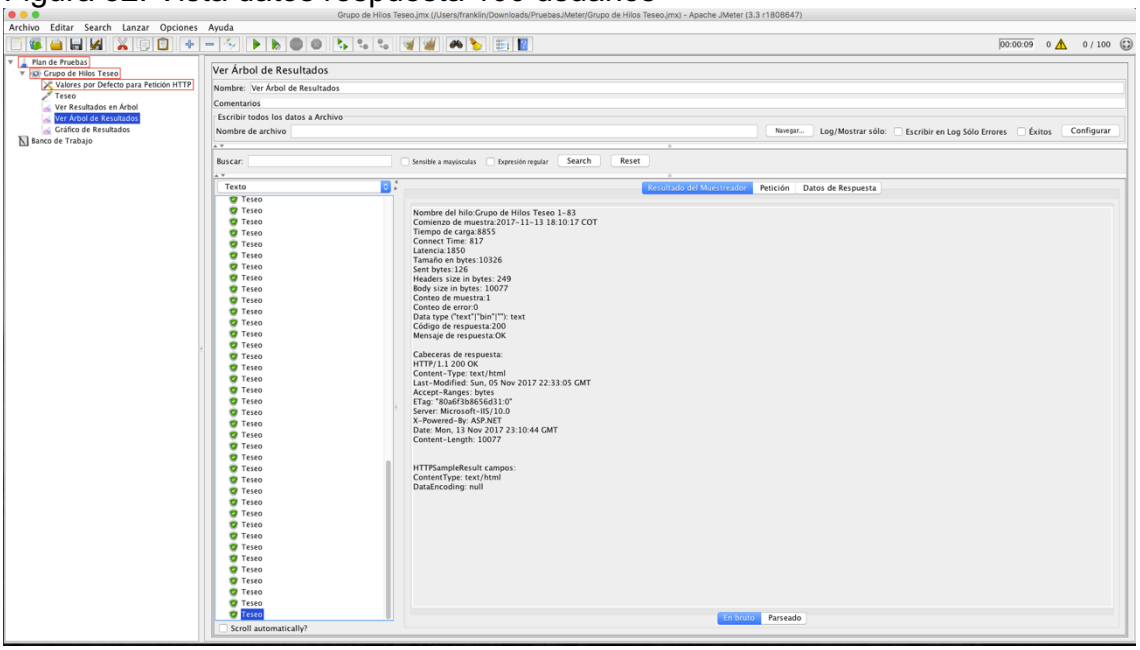
Autores

Figura 31. Vista datos exitosos 100 usuarios



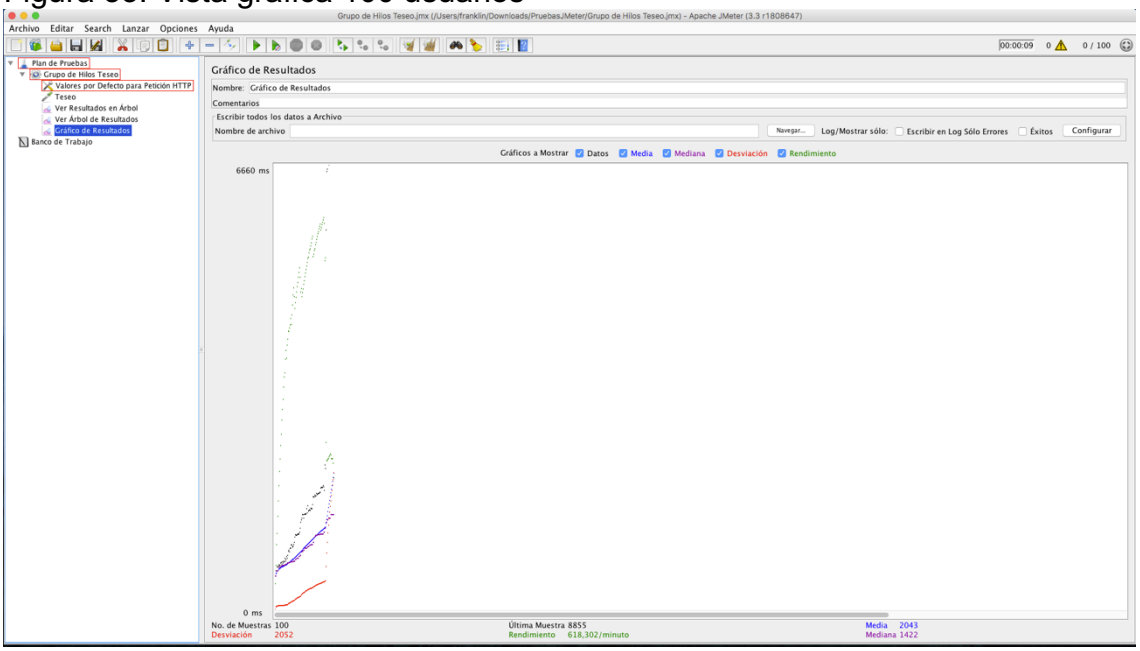
Autores

Figura 32. Vista datos respuesta 100 usuarios



Autores

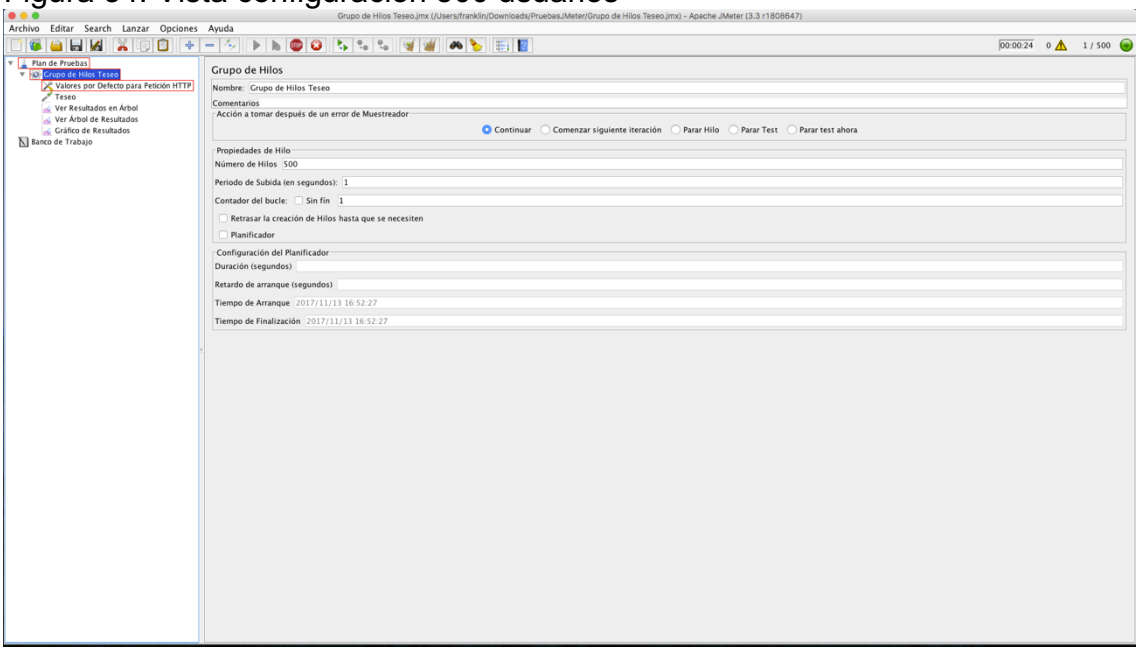
Figura 33. Vista grafica 100 usuarios



Autores

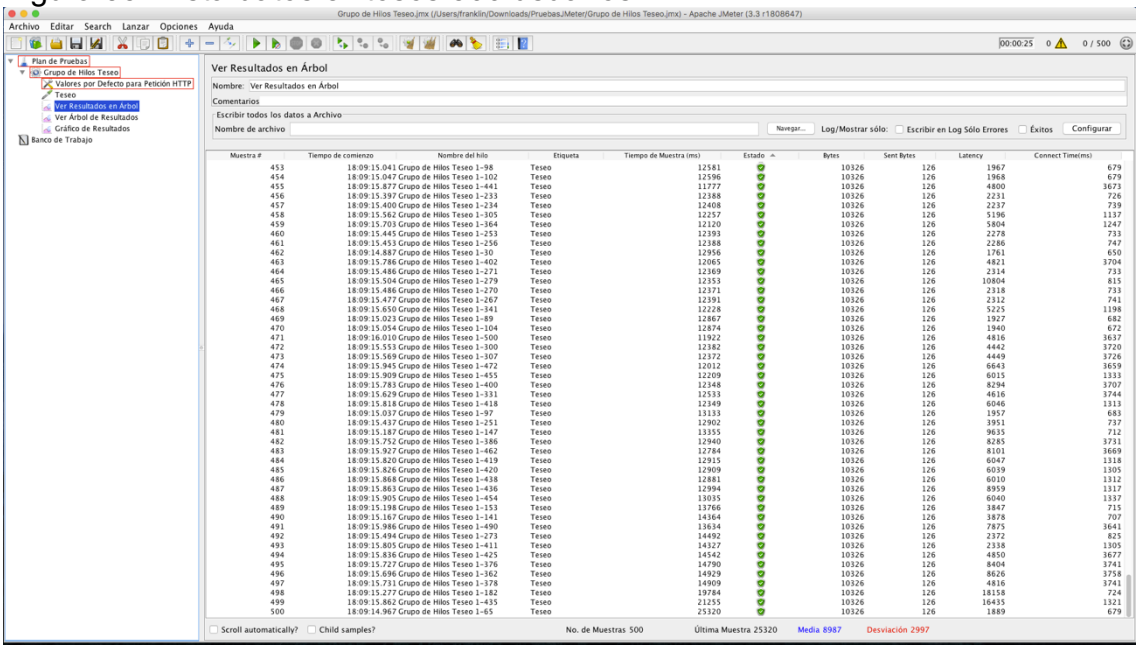
- Prueba 2 (500 Usuarios). Se realizo una prueba de que pueda responder a 500 usuarios sin errores y se cumplió satisfactoriamente.

Figura 34. Vista configuración 500 usuarios



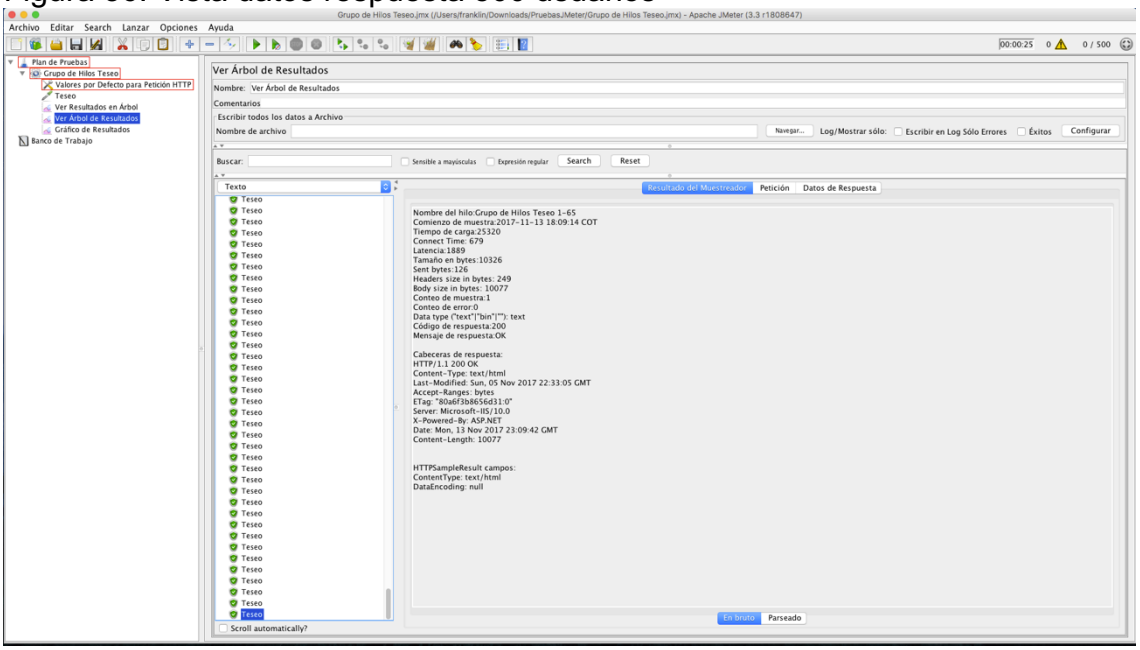
Autores

Figura 35. Vista datos exitosos 500 usuarios



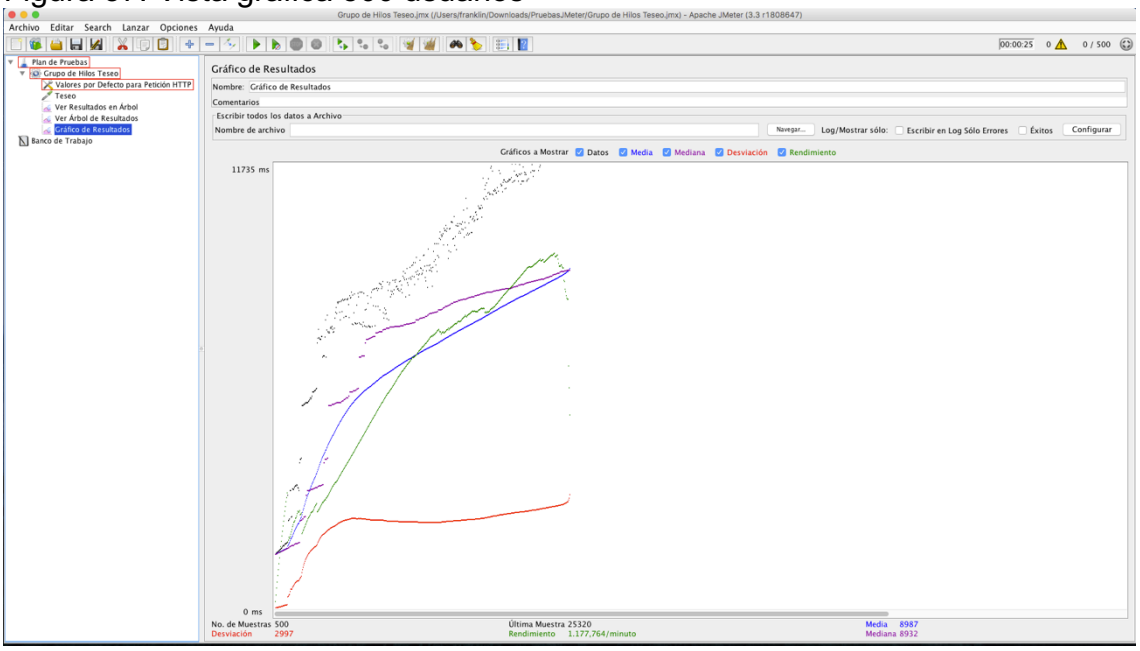
Autores

Figura 36. Vista datos respuesta 500 usuarios



Autores

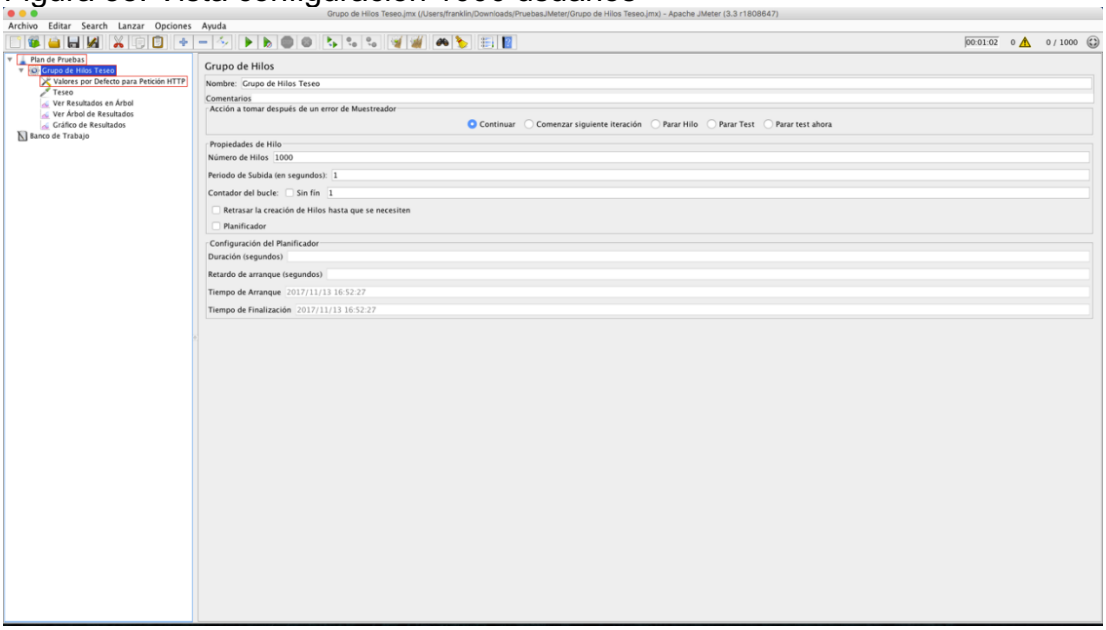
Figura 37. Vista grafica 500 usuarios



Autores

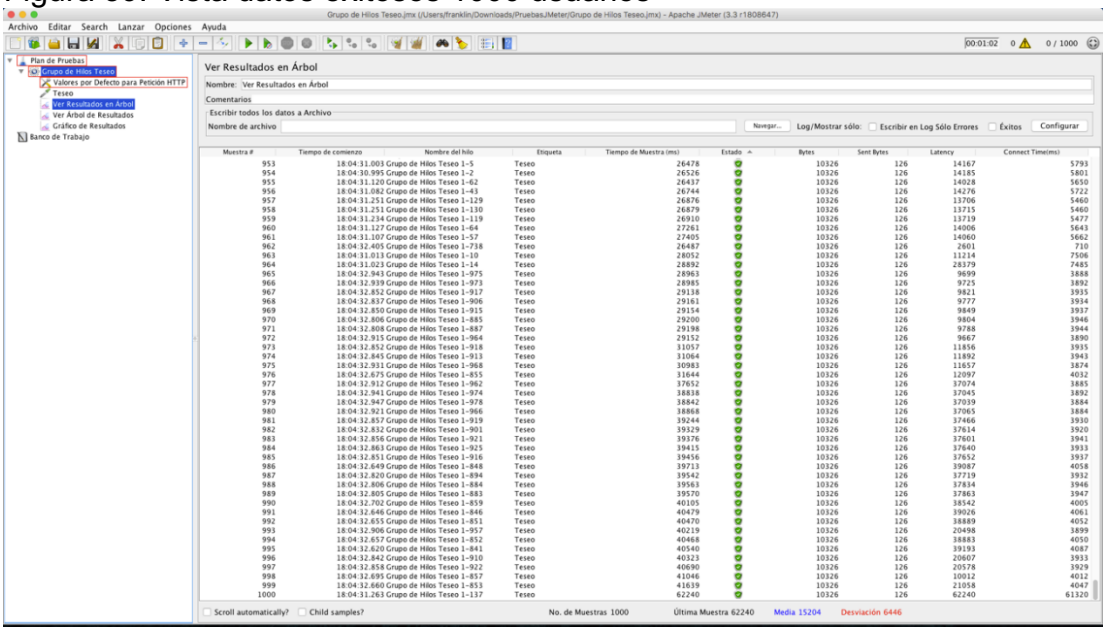
- Prueba 3 (1000 Usuarios). Se realizo una prueba de que pueda responder a 1000 usuarios sin errores y se cumplió satisfactoriamente, aunque se evidencia que el rendimiento bajo considerablemente.

Figura 38. Vista configuración 1000 usuarios



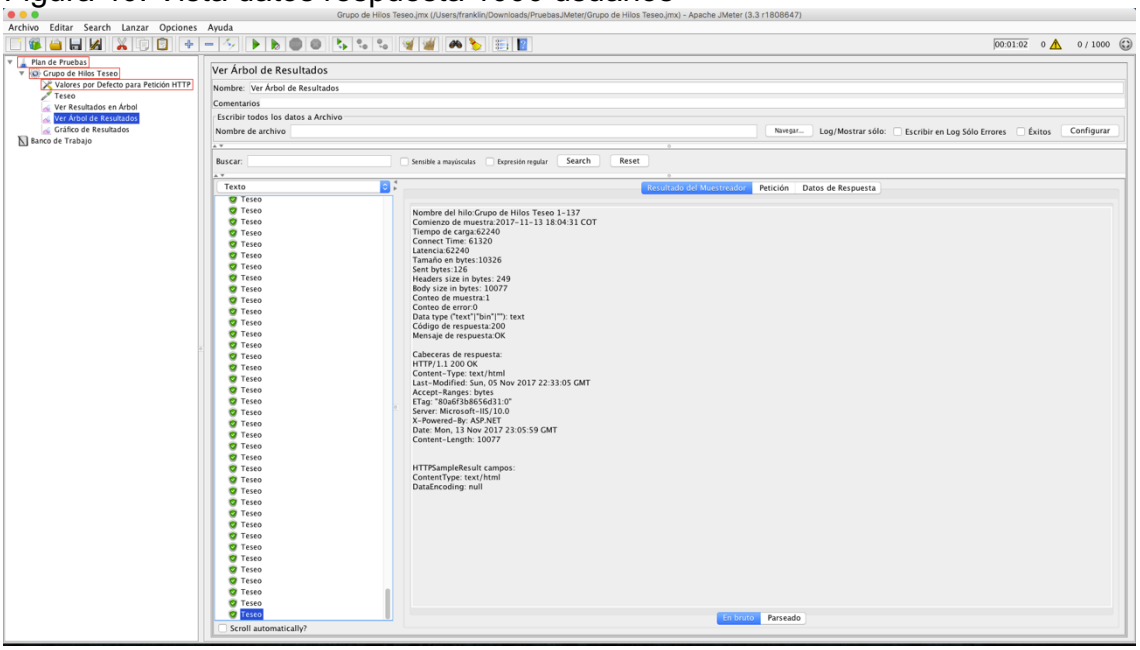
Autores

Figura 39. Vista datos exitosos 1000 usuarios



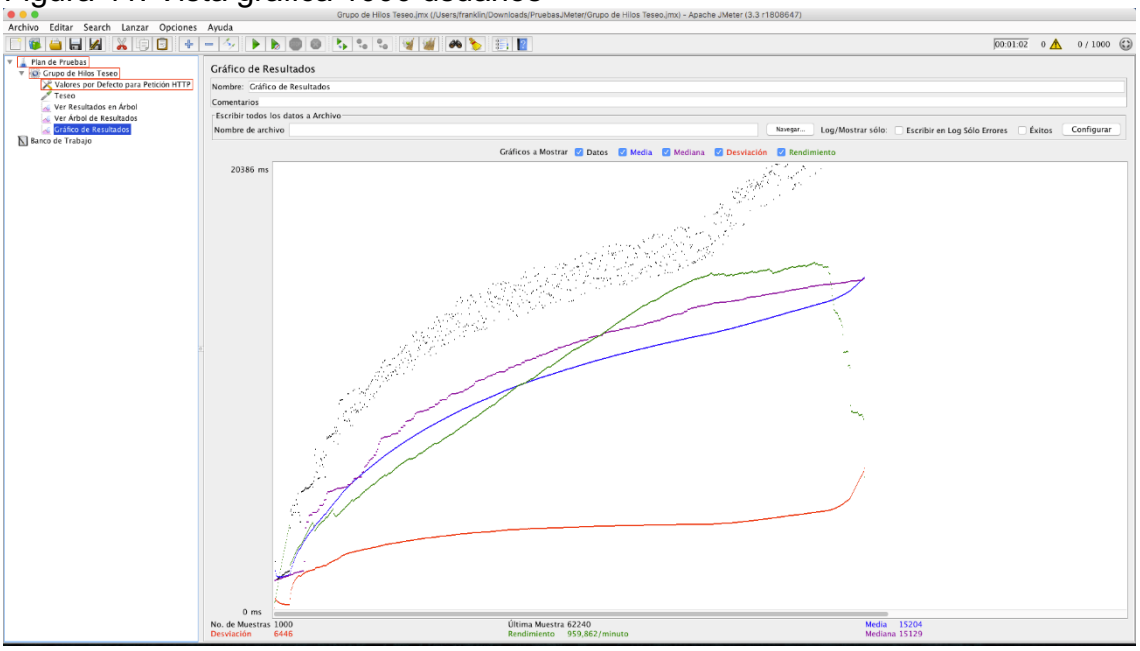
Autores

Figura 40. Vista datos respuesta 1000 usuarios



Autores

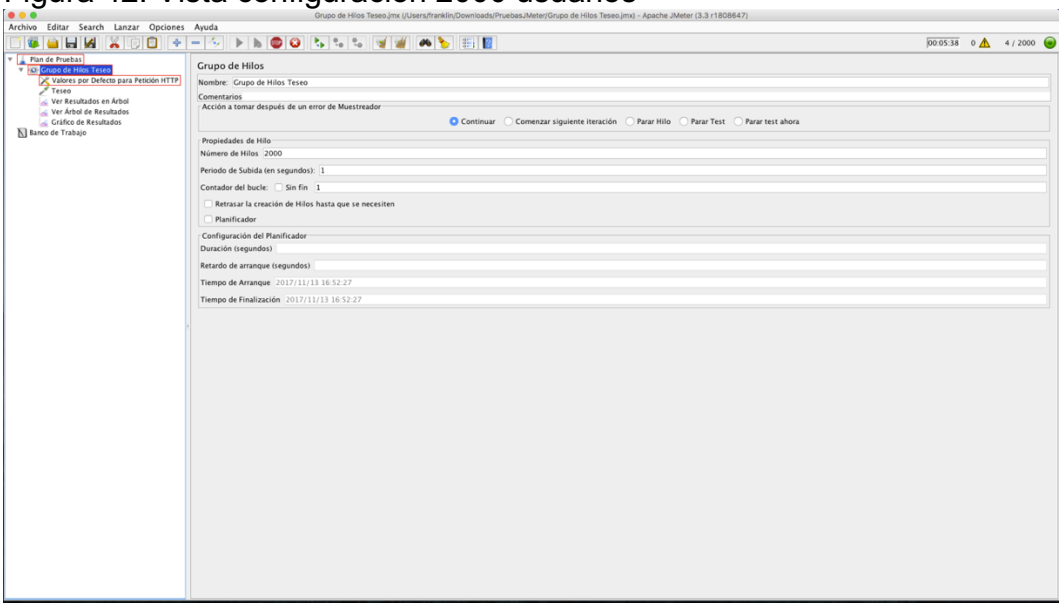
Figura 41. Vista grafica 1000 usuarios



Autores

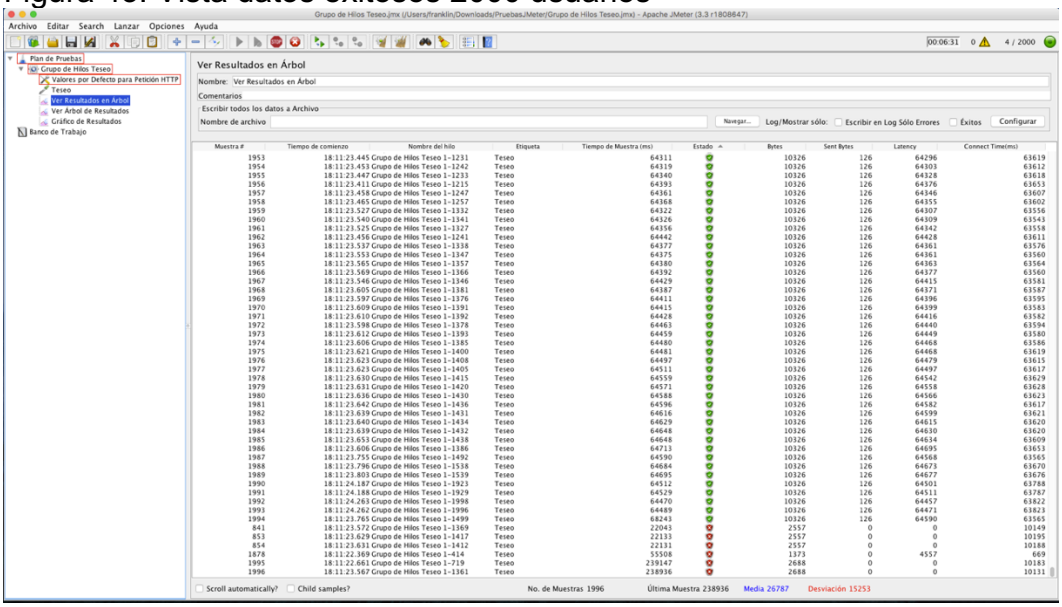
- Prueba 4 (2000 Usuarios). Se realizo una prueba de que pueda responder a 2000 usuarios sin errores y no se cumplió satisfactoriamente, aunque se evidencia que el rendimiento bajo considerablemente, también se ve que la aplicación soporta una carga de estrés de 1800 usuarios aproximadamente en línea.

Figura 42. Vista configuración 2000 usuarios



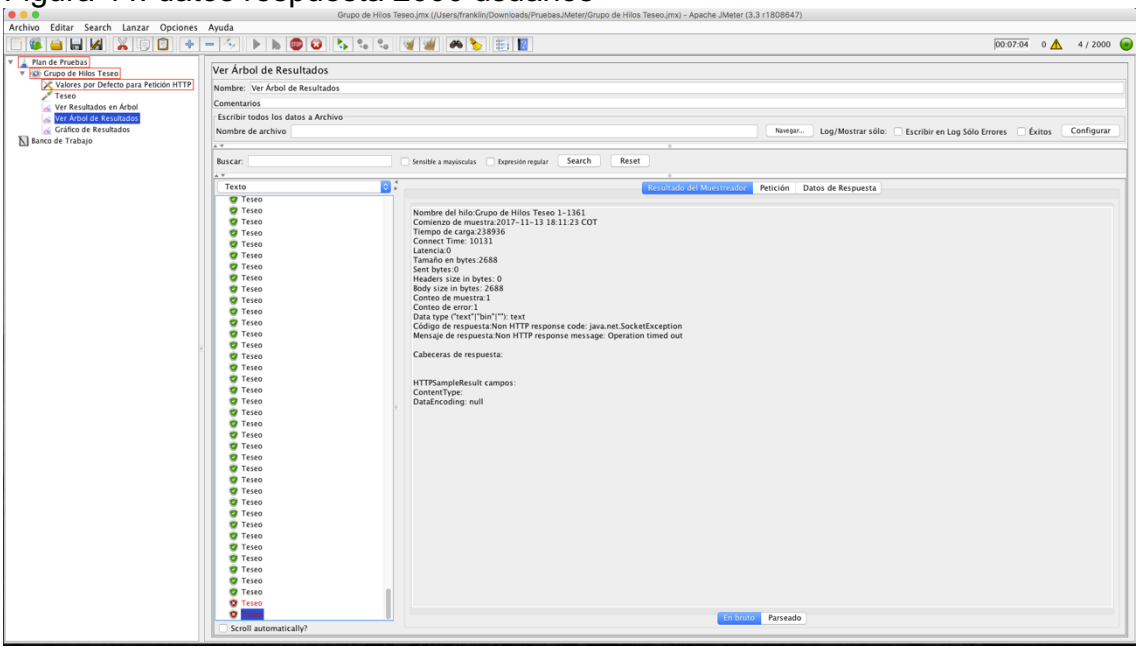
Autores

Figura 43. Vista datos exitosos 2000 usuarios



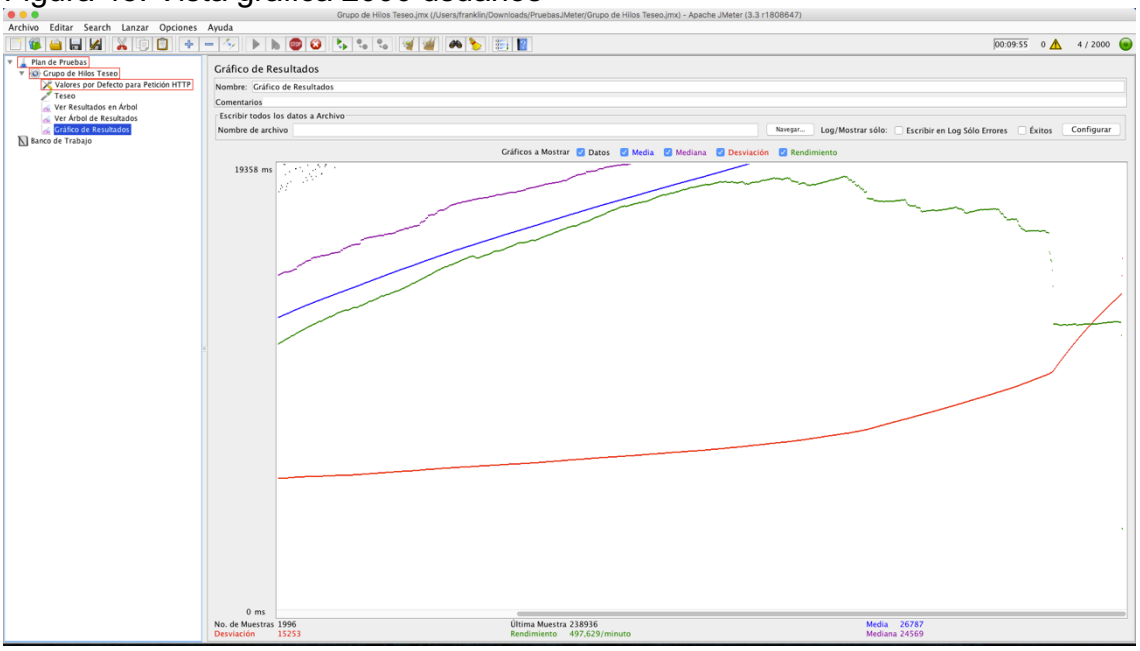
Autores

Figura 44. datos respuesta 2000 usuarios



Autores

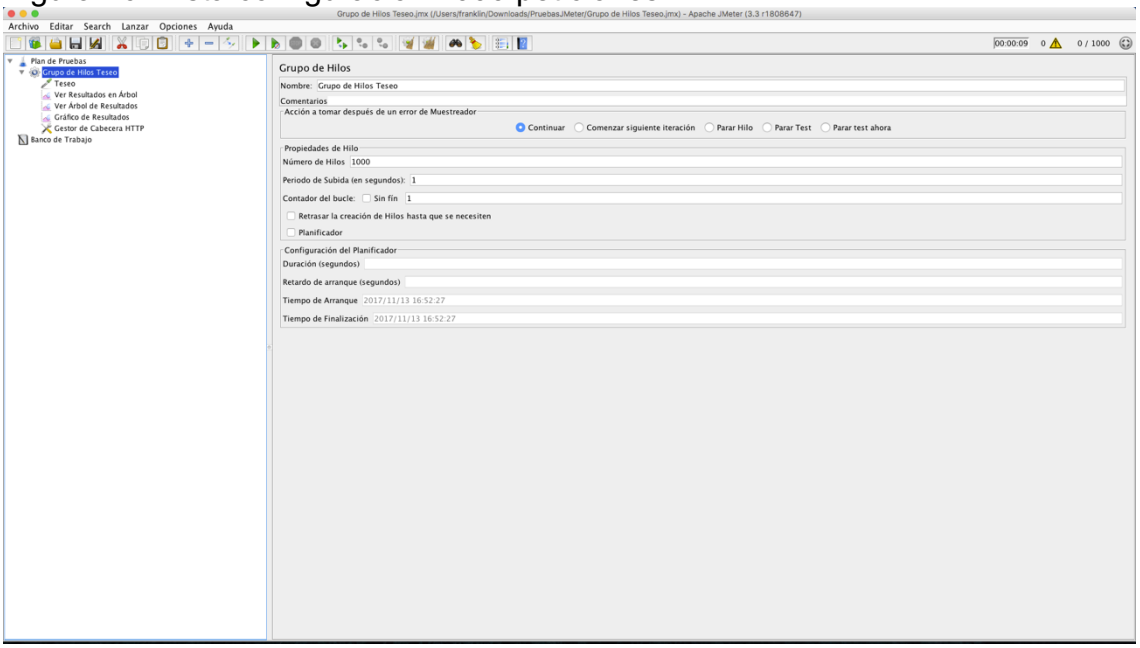
Figura 45. Vista grafica 2000 usuarios



Autores

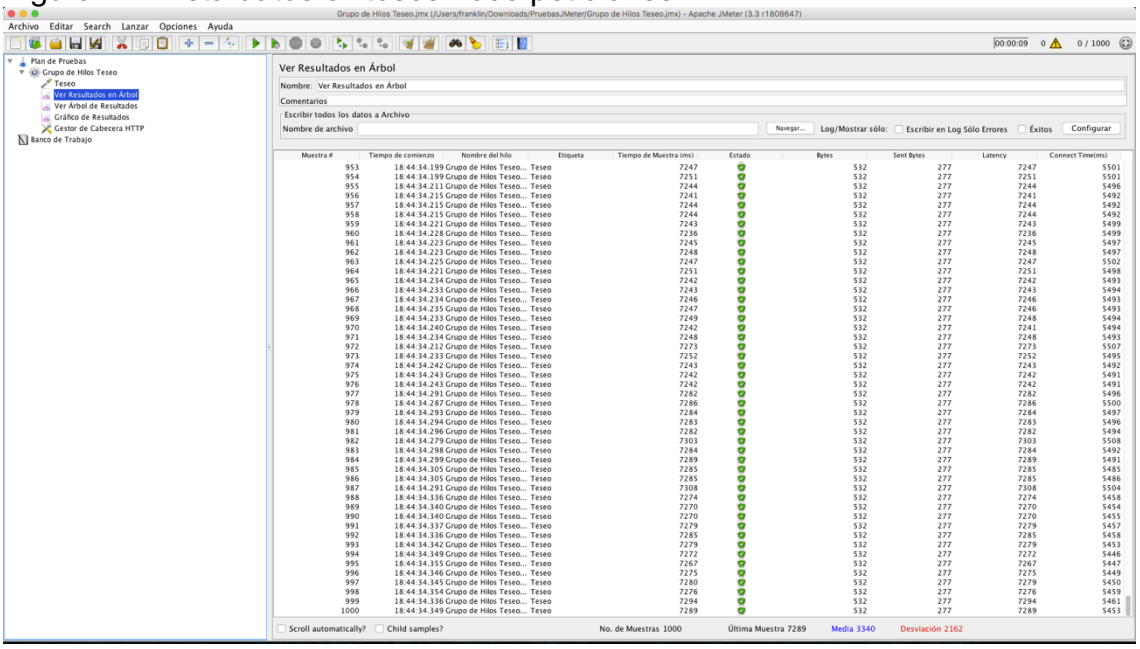
- Prueba 5 (1000 Peticiones). Se realizo una prueba a la capa de servicios de que pueda responder a 1000 usuarios sin errores y no se cumplió satisfactoriamente.

Figura 46. Vista configuración 1000 peticiones



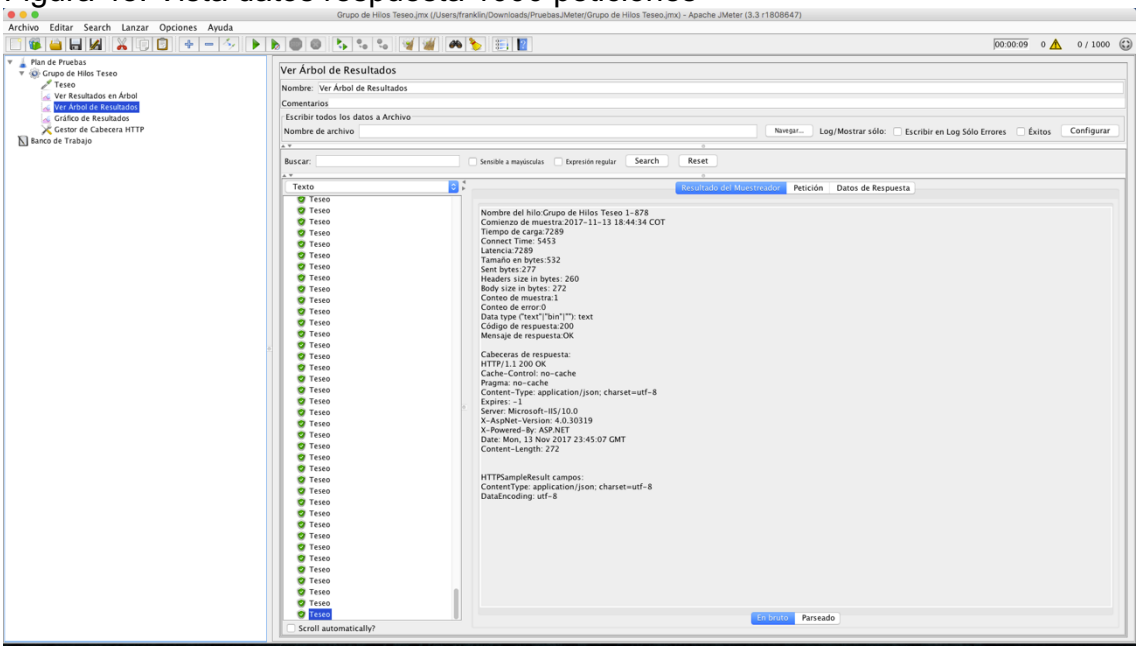
Autores

Figura 47. Vista datos exitosos 1000 peticiones



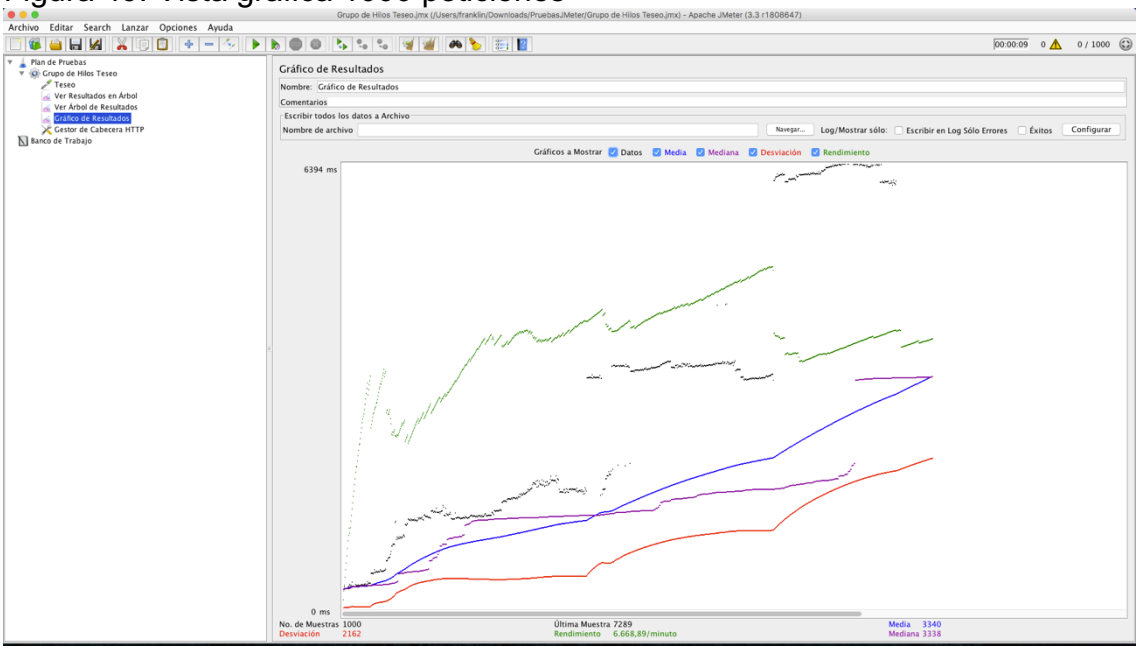
Autores

Figura 48. Vista datos respuesta 1000 peticiones



Autores

Figura 49. Vista grafica 1000 peticiones



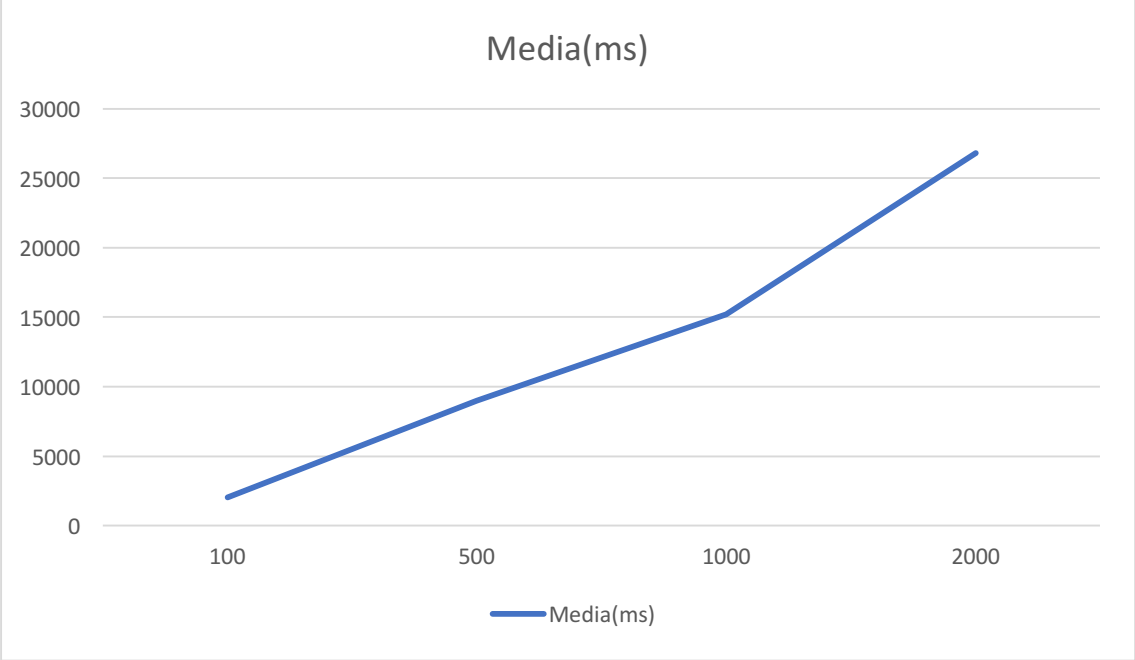
Autores

Tabla 37. Tabla comparativa de resultados de pruebas

Peticiones	Media(ms)	Desviación	# Fallos	Rendimiento por minuto
100	2043	2052	0	618,302
500	8987	2997	0	1177,764
1000	15204	6446	0	959,862
2000	26787	15253	6	497,629

Autores

Grafica 3. Resultados de prueba, Peticiones vs Media



Autores

A partir de la gráfica podemos determinar que el promedio máximo de peticiones para el servidor con el que contamos actualmente, el cual es un Windows Server 2012 R2 con 8GB en memoria RAM y procesador Core i3 de 4ª generación debe ser alrededor de 500 peticiones simultaneas, ya que después de 500 peticiones el rendimiento de la aplicación se ve seriamente afectado. Y después de 1000 peticiones no se puede garantizar que todas las transacciones finalicen correctamente.

10. DISCUSIÓN

Esta investigación tuvo como propósito identificar una mejor forma de enseñar a los estudiantes de programación de primeros semestres. Sobre todo, se pretendió examinar cuáles son aquellos problemas que más se presentaron al momento de iniciar en el mundo de la programación y cuál era la principal dificultad para una persona que pretende iniciar a programar sin muchos conocimientos previos. A continuación, se estarán discutiendo los principales hallazgos de este estudio.

De los resultados obtenidos en esta investigación, se puede deducir que la adaptación al manejo de un IDE complejo se reporta como uno de los problemas más comunes que tiene un estudiante de primeros semestres. Adicionalmente los errores sintácticos y gramaticales representan en gran medida una frustración y pérdida de tiempo para estudiantes que primero deben obtener una lógica clara de la estructura de un programa.

Por otro lado, de estos datos se puede deducir que, si un estudiante desarrolla la lógica computacional antes de enfocarse en aprender la sintaxis exacta de un lenguaje de programación, se pueden obtener resultados más rápidamente y un conocimiento más profundo del desarrollo de un programa.

Por otro lado, si comparamos los resultados con los encontrados en estudios realizados en la Universidad de Valparaíso, podemos observar que la deserción aproximadamente del 47% se produce en los primeros años de ingeniería.

Teniendo una cifra inquietante respecto a las asignaturas de programación en las cuales los estudiantes inscribían en promedio 5.14 y aprobaban tan solo 1.94.

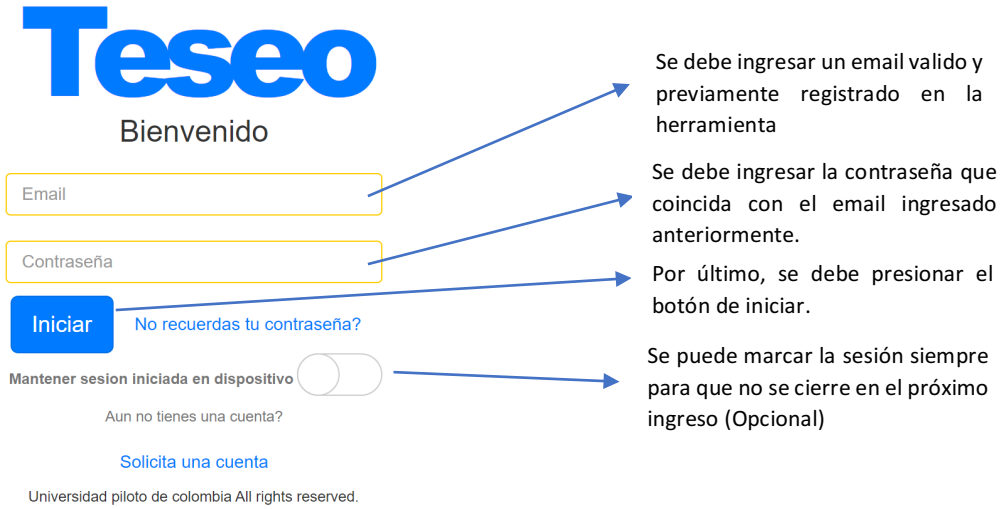
11. MANUAL DE LA APLICACIÓN.

Aquí se diseñó la parte del manual de la aplicación explicando detalladamente el funcionamiento de la aplicación.

11.1 INICIAR SESIÓN

El primer paso para utilizar la herramienta es iniciar sesión para ello es requerido un email valido y una contraseña.

Figura 50. Iniciar sesión



Autores

11.2 REGISTRO DE USUARIOS

Si se desea realizar el registro de un usuario nuevo se debe acceder a la opción de “Solicita una cuenta” y se ve una pantalla como la siguiente.

Figura 51. Registrarse

Autores

11.3 EDICIÓN DE CUENTA/PERFIL

Si se desea editar la información de la cuenta se debe presionar en el menú inferior “Mi Cuenta” y se debe desplegar una pantalla como la siguiente.

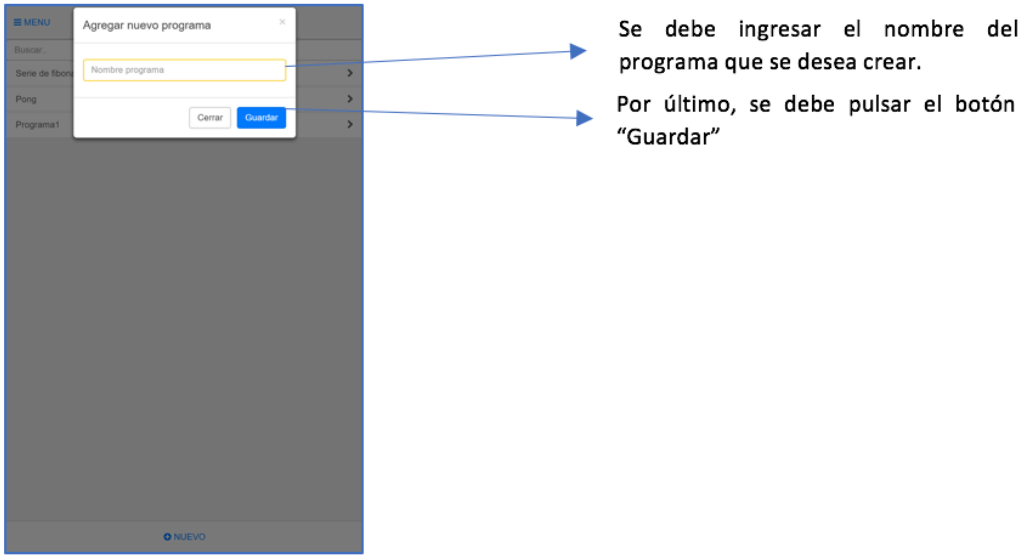
Figura 52. Mi cuenta

Autores

11.4 NUEVO PROGRAMA

Para crear un nuevo programa en el sistema se debe pulsar en la parte inferior el menú “Nuevo” y se desplegara una pantalla como la siguiente.

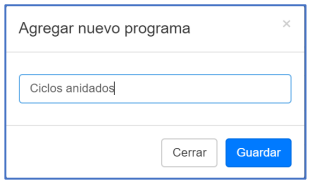
Figura 53. Nuevo programa



Autores

- Ejemplo:

Figura 54. Entrada de texto nombre

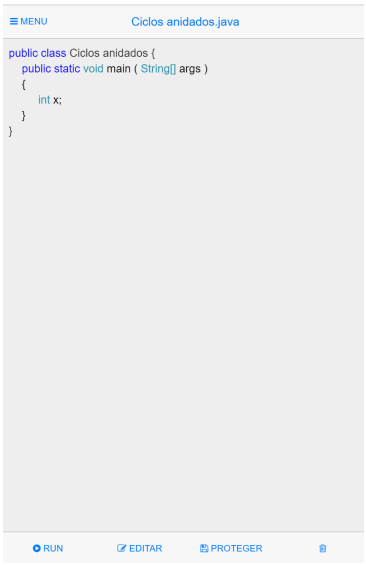


Autores

11.5 ESTRUCTURA BÁSICA

Una vez creado el programa la herramienta crea una estructura básica de programa y declara una variable por defecto a modo de ilustración.

Figura 55. Estructura Básica

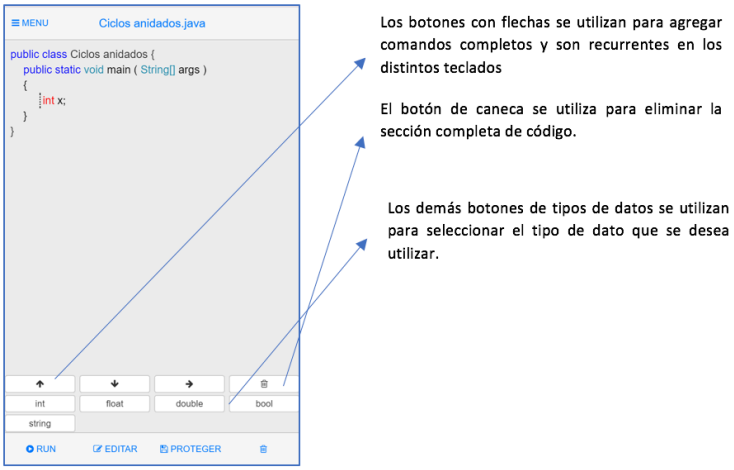


Autores

11.6 TECLADOS

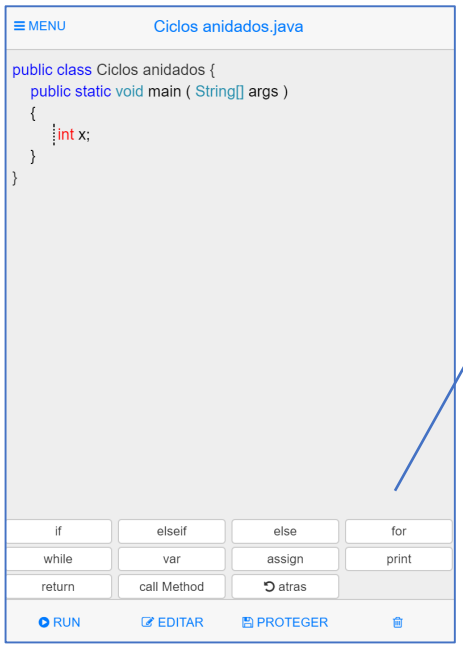
En la herramienta existen diversos tipos de teclados, el primero que podemos observar en la estructura básica es el de tipos de datos, para acceder a él se debe tocar un tipo de dato en el programa.

Figura 56. Teclado Estructura básica



Autores

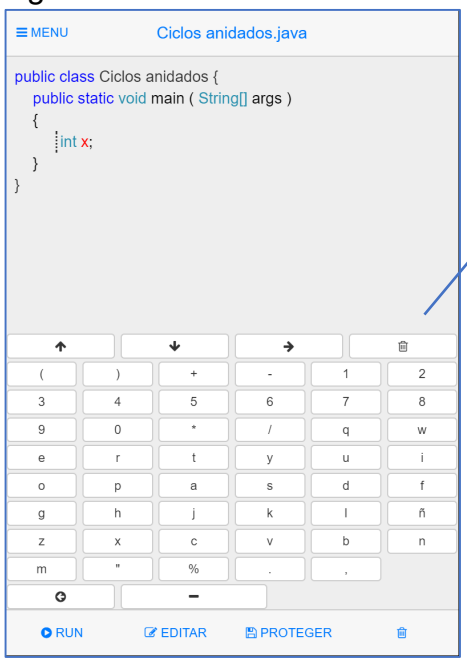
Figura 57. Teclados tipo de bloque



Una vez se presiona uno de los botones con flechas se desplegará un teclado que permite seleccionar el tipo de bloque a insertar.

Autores

Figura 58. Teclados libre



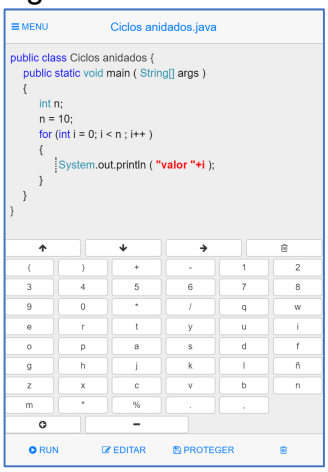
Otro de los teclados existentes es el general el cual permite escribir textos libres ejemplo para la creación de variables de texto, los nombres de métodos, parámetros y nombres de variables.

Autores

11.7 CICLOS

La herramienta cuenta con la opción de declarar ciclos para ello es necesario declarar uno de los tipos de ciclos y agregarle el comportamiento que se desea desencadenar en cada iteración, para el ejemplo siguiente declaramos una variable que contendrá el valor 10 y se ejecutara un ciclo que recorrerá la variable (Se ejecutara 10 veces) y en cada una de las iteraciones imprimirá el valor actual de la variable i.

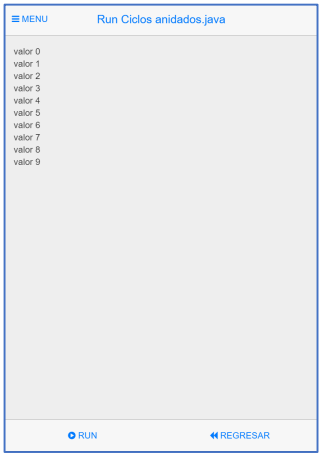
Figura 59. Ciclos



Autores

Resultado:

Figura 60. Resultado ciclos

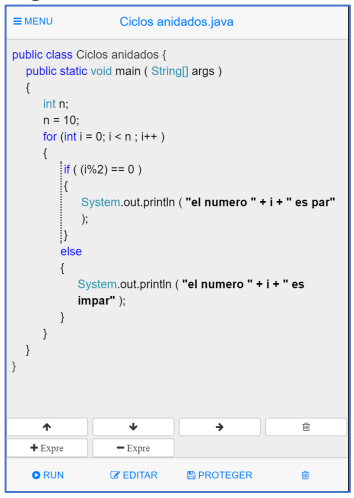


Autores

11.8 CONDICIONALES

La herramienta cuenta con la opción de declarar condicionales de tipo if, para ello es necesario declarar un condicional y agregarle una expresión lógica, en el siguiente ejemplo vamos a utilizar el ejemplo anterior y vamos a imprimir dos textos diferentes uno en caso de que el módulo sea par y otro en caso contrario.

Figura 61. Condicionales



Autores

Resultado:

Figura 62. Resultado condicionales

Autores

12. CONCLUSIONES Y RECOMENDACIONES

La herramienta permite a los estudiantes realizar el desarrollo de algoritmos básicos de una manera estructurada y ágil, también permite que los estudiantes de la universidad piloto puedan entrenar habilidades lógicas y algorítmicas en la programación.

La aplicación cuenta con una arquitectura cliente servidor, lo que permite que los programas creados por un estudiante puedan estar disponibles desde cualquier otro dispositivo ya que estos programas no se guardan localmente, si no en la nube.

La aplicación permite a los estudiantes realizar algoritmos básicos disminuyendo los errores sintácticos y gramaticales.

La aplicación permite a los estudiantes realizar programas desde cualquier dispositivo móvil que cuente con acceso a internet en cualquier momento, también permite a los estudiantes crear una cuenta personalizada y crear programas ilimitados asociados a la cuenta.

Para desarrollar una aplicación se debe tener en cuenta la usabilidad, ya que el desarrollo para varios dispositivos es muy complejo, puesto que las especificaciones de los proveedores de dispositivos cambian de acuerdo al avance de la tecnología, o por estrategia de mercado, esto conlleva a que no exista un estándar para el desarrollo de software.

Se desarrolló una herramienta que interpreta código en bloque a partir de una estructura JSON y que además cuenta con un teclado dinámico que agiliza la construcción de un programa. Además, la mejor manera de desarrollar la herramienta fue utilizando el Framework de Angular JS, ya que al utilizar las directivas permite hacer una recursividad en las vistas de cada primitiva, para así poder conformar el árbol de la estructura del programa y de esta manera hacer el 100% de reutilización de código.

13. BIBLIOGRAFÍA

AHO, Alfred. et al. Compiladores: principios, técnicas y herramientas. 2ª Edición. México: Pearson Educación, 1990. 820p.

BELLER, Walter. Érase un tópico en filosofía: la lógica dialéctica. En: Revista Casa del tiempo. Julio-Agosto, 2015, Vol. 2, N 18-19, p. 32-34.

BEUCHOT, Mauricio. Introducción a la lógica. México D.F: Universidad Autónoma de México. 1ª edición, 2004. 175p.

CELA, Karina. et al. Evaluación de herramientas web 2.0, estilos de aprendizaje y su aplicación en el ámbito educativo. En: Revista Estilos de Aprendizaje. Abril, 2010, Vol. 3 N.5, p. 117-134.

DEITEL, Harvey y DEITEL, Paul. Como programar en java. México: Pearson Educación. 9ª Edición, 2012, 616p.

DELGADO, Francisco y AMADOR, César. Algoritmos resueltos con diagramas de flujo y pseudocódigo. México: Universidad Autónoma de Aguascalientes. 1ª Edición, 2014. 170 p.

GARCÍA, José Luis y GARCÍA, Rosa. Aprendizaje entre Iguales con Herramientas Web 2.0 y Twitter en la Universidad. Análisis de un Caso. En: Revista Electrónica de Tecnología Educativa. Junio, 2012, Vol. 40, p. 1-14.

JUGANARU MATHIEU, Mihaela. Introducción a la programación. México D.F: Grupo editorial patria, S.A de C.V. 1ª Edición, 2014. 320p.

LEFEBVRE, Henri. Lógica formal, Lógica dialéctica. España: Siglo XXI Editores. 1ª Edición, 1993. 346p.

LÓPEZ, Juan Carlos. Algoritmos y Programación. En: Fundación Gabriel Piedrahita Uribe. 2ª Edición, 2009. 96p.

MANCILLA, Alfonso, EBRATT, Jesús y CAPACHO, José. Diseño y construcción de algoritmos. Colombia: Editorial Universidad del Norte. 2014. 414p.

MICROSOFT, Convenciones de código de C# (Guía de programación de C#) [En línea], Microsoft, [citado el 1 de noviembre de 2017], disponible desde: https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/inside-a-program/coding-conventions.

MINTIC, (Suscriptores a internet fijo y móvil en: Boletín trimestral de las TIC) [en línea]. (Colombia): Ministerio de tecnologías de la información y las comunicaciones Julio, 2017- [citado 1 de septiembre], disponible desde: http://colombiatic.mintic.gov.co/602/w3-article-55212.html.

NOVACK, George. Introducción a la lógica, lógica formal y lógica dialéctica. España: Editorial Fontamara, S.A. 1ª Edición, 1979. 74p.

PROGRAMACIÓN DE COMPUTADORES. Publicación Facultad de Ingeniería, Universidad Nacional de Colombia, 2004.

RUIZ CATALÁN, Jacinto. COMPILADORES: Teoría e Implementación. España: RC Libros, 2010. 16p.

SOMMERVILLE, Ian. Ingeniería del software. España: Pearson Educación, S.A. 7ª Edición, 2005. 712p.

VAQUERIZO, Belén, RENEDO, Eduardo y VALERO, Miguel. Aprendizaje colaborativo en grupo: Herramientas Web 2.0. En: XV JENU. Julio, 2009, p. 447-450.

VELARDE, Julián. Lógica y Dialéctica 1. En: Teorema: International Journal of Philosophy, 1977, Vol 7, No 2, p. 129-140.

WOODS, Alan y GRANT, Ted. La lógica formal y la dialéctica. En: Razón y Revolución. Reedición Electrónica, 2002, No 10, p. 1-27.